MapInfo

MapXtreme Java 版

开发人员指南

MapInfo Corporation
Troy, New York

MapInfo 公司总部：
语音电话：(518) 285-6000
传真：(518) 285-6070
销售信息热线：(800) 327-8627
政府机构销售热线：(800) 619-2333
技术支持热线：(518) 285-7283
技术支持传真：(518) 285-6080

欲获取 MapInfo 公司的联系信息，也可查询以下网页：**http://www.mapinfo.com/contact_us.cfm** 。

2003 年 9 月

目录

1 简介

MapXtreme Java 版是企业级的地图绘制开发工具，可实现数据的可视化和地图绘制，帮助企业作出更加出色的决策，并更加有效地进行运营和管理资产。在可管理的服务器网络上实现应用程序的运行，不仅在很大程度上提高了规模效益，如降低硬件和管理成本，同时还明显改善了应用程序性能、可靠性和安全性。使用 MapXtreme Java 构建的应用程序既可适用于企业内部网，也可适用于公共的因特网。

本章内容：

产品组件

MapXtreme Java 是一个 100% 的纯 Java 类 （Java 2 兼容）集合，借助于这一集合，可将应用程序部署到各种系统，无论是 Windows、 UNIX 系统，还是这两种系统的混合环境。本产品包括以下组件：

- MapXtremeServlet 地图绘制引擎
- MapJ API 和 Java 类库
- MapXtreme JavaBean
- MapXtreme Java 管理器和 Web Application Builder
- JSP 定制标记库
- Java 2 VM (1.4.1)
- 示例应用程序
- 示例数据
- MapXtreme Java 文档集 （本开发人员指南、对象模型张贴画和完整的 MapJ API Javadoc）

产品目标用户

本产品面向开发人员设计，适用于创建集成动态地图以及与相应地图交互所需工具的 Web 应用程序。

对于经验丰富的 Java 程序员而言，可选择使用 MapJ API 通过编程设计定制的地图和功能。

对于 Java 程序员新手，则可选择以 JavaBean 形式提供的地图绘制功能，在可视化的 IDE 中通过鼠标的拖放，轻松实现开发。

即使不是程序员，也可使用 Web Application Builder 迅速创建 Web 应用程序原型，无需编程。 Web Application Builder 提供了逐个步骤的指南，帮助用户创建基于 Java 服务器页标记的地图绘制应用程序。

特性

MapXtreme Java 是基于 Java 2 企业版中指定 servlet 体系结构的企业级应用程序开发工具。MapXtremeServlet 是本产品的关键组件之一，用于在服务器端管理地图绘制服务的请求和响应，其中包括图像请求 (GIF、 JPEG)、向量数据请求 （查询方法）和元数据请求（表信息）。

借助于 servlet 体系结构， MapXtreme Java 可以侧重于处理地图绘制请求，而由 Web 服务器 /servlet 容器处理例如负载平衡、安全和容错等其他服务器端问题。此外，相应 servlet 模型使用 HTTP 这一因特网上的标准通信协议。 MapXtreme Java 的其他特性包括：

- 诸如选择行为、专题地图绘制和分析以及高级标注和渲染等功能的高级地图绘制功能。
- 通过 JDBC 访问数据源，不仅可在安全 RDBMS 上维护空间数据，同时还可利用其全部潜力构建地图。
- 轻松便捷的安装、配置、开发和部署，本产品提供了 JavaBean、 Web Application Builder、 JSP 定制标记库、示例应用程序和 Apache Web Server/Tomcat servlet 容器等众多便利。

优点

MapXtreme 提供了支持多平台、高质量、高性能的易用解决方案来满足各种地图绘制需求。

多平台支持

出于安全性、可靠性和性能方面的种种原因，诸如电讯和保险业等众多面向最终用户的行业，均采用了 UNIX 系统；而与此同时，同一企业中的不同用户，还有可能使用 Windows 系统。在多平台上部署类似解决方案是常见的需求之一。借助于基于 Java 的地图绘制应用程序，开发人员编写一个程序，即可在支持虚拟机的多种平台上运行。

将 MapXtreme Java 部署在服务器端运行，即可充分利用现有的 UNIX 或 Windows 资源。开发人员可以在一个系统上存储和控制数据，同时通过编程从其他运行虚拟机的计算机实现相应数据的访问。

高扩展性

对于使用 MapXtreme 创建企业级地图绘制解决方案的组织而言，所开发的应用程序不仅应该可以良好运行，而且还应该可以支持需要访问各种应用的所有用户。 MapXtreme 为此采用基于组件的策略，提供了可靠的多线程解决方案，确保了出色的适应性和扩展性。借助于此，企业即可根据组织的具体需求来扩展应用程序。

快速部署

借助于 MapXtreme，用户可以实现应用程序的快速开发、安装和部署。本产品为此提供了用于快速开发原型的 Web Application Builder 向导，以及展示 MapXtreme Java 基本概念和知识的若干示例应用程序。这些示例应用程序既可以进一步开发为专用的应用程序，也可以作为用户自行开发应用程序所依赖的基础。 MapXtreme Java 与所有 J2EE 验证的 Web 服务器 / 浏览器兼容，并且不使用专属的插件。

编程优点

面向对象

MapXtreme 是面向对象的开发工具，采用易用的对象模型分层结构，用于地图的显示、查询和分析。

MapJ API

MapJ API 是用于与地图绘制引擎 MapXtremeServlet 通信的客户端 API。每个源自 MapXtreme 的客户端请求地图均使用 （或复用） MapJ 对象的实例。由于 MapXtreme 没有采用专属的插件，因此其可以将地图传输到任意操作系统上的任意浏览器。

MapXtreme 是异步多线程和无状态的，实现了性能的最大化。

服务器端 Java

大部分 Web 应用软件提供商的原始解决方案均为全状态的客户端工具，这些工具必须与特定的服务器环境相匹配。与此相反， MapXtreme 从设计之初就是服务器端的 Java 组件。使用 MapXtreme 开发的应用程序尤其支持：

- 众多并发用户
- 计算机群集

- 采用多 CPU 的服务器
- 任意平台 （包括 Windows NT 和众多版本的 UNIX）
- 数据库连接池
- 将安全问题与客户端实施相隔离

借助于此， MapXtreme 基于 Web 的地图绘制应用程序可扩展以支持众多用户，并可由应用服务器来管理相应的用户。 MapXtreme 可以与采用类似 Apache/Tomcat、 JRun 或 IBM WebSphere 的 Web 服务器的高负载网站高效协同工作。

智能多线程

MapXtreme 使用由 servlet 容器 / 应用服务器提供的智能 Java 线程来有效处理多个并发用户的负荷。这样不仅消耗的内存较低，而且还可以通过添加额外的 CPU 来支持更高的用户负荷。测试表明， MapXtreme 服务器引擎在稳定状态下需要约 8MB 内存，另外每个并发用户增加 100 KB 至 200 KB 内存。例如，假定一个线程正在处理地图请求。那么与此同时，另外三个线程可以通过网络 I/O 来传递此前三个地图请求的结果。

基于组件的灵活性

MapXtreme 采用基于组件的体系结构，在部署上实现了相当高的灵活性。本产品共有 4 个高层组件，如下所示：MapJ 对象、显示地图的地图渲染器、访问各种数据源的数据提供方以及 MapXtremeServlet。 MapXtreme 可用于两层的内部网部署，将 MapJ 置于客户端；也可用于因特网的三层配置，将 MapJ 和业务逻辑置于中间层。

针对远程数据集的强大连接性能

MapXtreme 顺应在关系数据库中存储空间数据的潮流，提供了对于带有空间选项的 Oracle 和带有 SpatialWare DataBlade 的 Informix Dynamic Server 等数据库管理系统的支持。借助于此，即可在企业级数据库管理系统中维护空间数据，确保关键任务的完成，也可同时为万维网上的任意用户提供适当的访问权限。

兼容任意 Web 环境

MapXtreme 的开放式体系结构完全兼容几乎所有的 Web 环境 （尤其是三层体系结构），并可以和支持 ISAPI、NSAPI 或 CGI 网关的任意 Web 服务器系统（如 Netscape、 Apache）工作。作为一个 servlet 环境， MapXtreme 兼备了 Sun 的 Java Servlet API 的全部优点。

MapInfo 建议的环境体系结构包括可以生成 Java 对象实例的应用服务器。为了便于使用，产品光盘上提供了 Apache Web Server 和 Tomcat servlet 容器。

事实上，任意 Web 浏览器均接受 MapXtreme 生成的地图，因为 MapXtreme 可以将地图图像作为 GIF 或 JPEG 等栅格图像输出，这些图像随后接口嵌入到 HTML 中。有些功能更强大的浏览器 （或带有 1.2 或更高版本 VM 插件的浏览器）还可以接受向量数据，并使用这些向量显示地图。 MapInfo 建议使用 Netscape 或 Internet Explorer 4.x 或更高版本。

便于编程

MapXtreme Java 提供了众多工具和便利，协助用户开发 Web 应用程序。 MapXtreme Java 管理器提供 Web Application Builder 向导，引导您创建 Web 应用程序。这对于快速创建体现概念应用程序的原型尤为实用。该向导使用 Java 服务器页技术，采用定制 JSP 标记的形式，简化了地图绘制应用程序所需的必要编码。 MapXtreme 还提供了众多 JavaBean 以及可拖放至可视化 IDE 界面中的地图绘制组件。

使用 MapXtreme Java 可以做什么？

使用 MapXtreme Java 可以构建两层和三层 Web 地图绘制程序，处理源自客户机的地图数据请求。借助于其灵活的可扩展体系结构，可根据具体绘图需要，向客户机发送软件、控制对于敏感数据的访问和随需求的增长来扩展应用程序。

程序员尤其可以：

- 设计只具备所需的特性和信息的定制地图
- 通过程序创建静态和动态对象
- 自定义图元的外观、位置和行为
- 监听类似用户鼠标点击的地图事件，初始化地图的变更

对于要使用地图绘制应用程序的最终用户而言，可为其提供工具来实现：

- 缩放和平移，以更改地图视野
- 选择图元并绘制搜索区域
- 查询图元以获取更多信息
- 创建专题影线表示，例如基于数据库数据的颜色编码地图
- 控制对象和标注的可见性和样式

实用

MapXtreme Java 在提供众多强大功能的同时，还可以通过若干种方式，将地图绘制功能集成到企业应用程序中。

- 为现场技术人员提供实时访问公司和客户数据的能力，更好地从现场提供服务。
- 在电信和运输领域，地图提供了一种监视网络的途径，用于确定问题区域、性能瓶颈或维修状态。
- 在供应链管理中，地图绘制应用程序可用于查看货物、服务或人员的分布，并根据需要重新分配相应资源。
- 创建用于供货卡车、网络电话或军队部署的跟踪应用程序。
- 实现更加便捷的客服自助服务应用，例如用于公众访问有关政府计划的信息。

MapXtreme Java 文档集

本开发人员指南是开始学习 MapXtreme Java 的最佳资源。此外，还有若干附加信息资源对于 MapXtreme 开发人员非常实用，其中包括：

- MapJ API 规格 (HTML) — 随 MapXtreme 安装的 Javadoc 提供
- MapJ 类和方法的对象模型张贴画
- Web 上的 MapXtreme Java — 网址为 **www.mapinfo.com/mapxtreme** 的 Web 资源，包括可访问的论坛和知识库

MapXtreme Java 4.7 版新增特性

MapXtreme Java 版的部分新增特性包括：

TAB 库

现在可以编辑本地附带的 MapInfo TAB 文件。有关详细信息，请参阅第 223 页第 13 章的*图元和搜索*和第 243 页的*编辑 Tab 图层*。

缓冲

MapXtreme Java 提供了用于创建缓冲区的新方法。可从任意输入几何对象（点、线或区域）创建缓冲区。有关详细信息，请参阅第 225 页的*缓冲*。

嵌入式栅格数据 URL

目前 Rendition.SYMBOL_URL 属性可支持嵌入式栅格数据 URL。借助于这一特性，可将图像嵌入到样式中，令样式和文档的传递更加便捷。有关详细信息，请参阅第 266 页的*图像符号*。

渐变填充和单笔填充

目前，可设置样式来使用渐变填充和单笔填充。可选使用线性或半径颜色渐变。线性渐变沿直线通过一系列颜色过渡。半径颜色渐变沿圆通过一系列颜色过渡。有关详细信息，请参阅第 260 页的*渐变*。

JPEG2000 支持

目前， MapXtreme Java 可加载在 TAB 文件中注册的 JPEG2000 栅格文件。支持所有栅格样式。其中包括亮度、对比度、灰度、透明度和半透明度。

图形用户界面新增特性

MapXtreme 4.7 提供了众多使用性能改进。其中包括：

- 在 MapXtreme Java 管理器客户机中，现在可以使用按键组合来切换选项卡。例如，使用 Alt-M 可以切换到 “地图定义”选项卡。
- 基于文件图层类型（Tab、 GeoTIFF、形状）的 “增加图层向导” GUI 的功能现已增强，可允许在一个向导中，从单独的目录选择多个文件。即运行向导一次，即可从各种目录中加载 10 个图层。
- MapXtreme Java 管理器客户机现在可采用单机模式运行。这意味着可以不启动服务器就运行客户机。某些功能在采用单机模式运行时将不可用，其中尤其包括 Web Applications 构建器向导。
- MapXtreme Java 管理器客户机现在包括第 4 个用于连接管理器的选项卡。用户借此即无需运行单独客户机应用程序来定义连接。连接管理器选项卡上的界面和上一版本的连接管理器相同。
- MapXtreme Java 管理器客户机现在支持右键单击菜单，借助于此，现在还可以直接在 VisualMapJ 中构建标准的右键单击菜单。
- 目前在开发应用程序时，可以限制 “图层控制”对话框中的缩放图层选项。可在 LayerControl 中使用 setAllowZoomLayerEditing 方法，禁用缩放图层选项。有关详细信息，请参阅 HTML API。

自然中断范围

在创建范围专题地图时，现在可选择使用自然中断范围的选项。自然中断范围包含分布值，以便每个范围的平均值可以和该范围的每个值尽可能地接近。有关详细信息，请参阅第 281 页的*分布类型*。

饼图和条形图

MapXtreme Java 提供了集成饼图、并行条形图和堆叠条形图的功能。有关详细信息，请参阅第 288 页的*使用分析图层*。

栅格图层优化

MapXtreme Java 现在可以采用更高的效率渲染栅格图层。栅格图层的性能、外观和效果均有明显改进。尤其体现在以下栅格样式中：亮度、对比度、灰度、透明度和半透明度。

MapXtreme 使用 Java 的高级成像 (JAI) API 和 Java 2 平台 SDK 1.4 的新图像 I/O (IIO) API 的组合，实现了众多新增特性。

样式选择器改进

样式选择器对话框可用于选择显示属性，例如线条图案和前景色。 MapXtreme Java 现在可以为创建和编辑命名样式以提供更多支持。主要的增强特性如下所示：

- 样式选择器可供多处使用。尤其重要的是，专题地图绘制对话框现在包括启动选择样式对话框的按钮，以便于指定颜色之外的更多属性。
- 如果从命名资源选项卡启动选择样式对话框，还可以保存新的命名样式。例如，如选择标准样式 （如 “铁路”直线样式），然后通过更改颜色、线宽等对其进行自定义，即可将所选样式另存为类似 “收藏 / 我的铁路”新命名样式。
- 除了前景色之外，现在还可以设置背景色。
- 通过使用滑块，现在还可以设置样式的不透明性。
- 可选定制 （GIF 或 JPG）图像来用作符号。

有关使用样式的详细信息，请参阅第 61 页第 5 章的*管理 MapXtreme Java* 。

可扩展的向量图形 (SVG) 输出

MapXtreme Java 版支持采用 SVG 格式导出地图图像，此图形格式用于在 XML 中描述二维图形。有关详细信息，请参阅第 205 页的 *SVG 输出* 。

SVG JSP 标记

现在，可使用 MapXtreme Java 定制 SVG JSP 标记，该标记也支持以 SVG 格式导出地图图像。有关详细信息，请参阅第 347 页附录 A 的*定制 JSP 标记库* 。

Web 地图服务

MapXtreme Java 目前还允许创建 OGC 兼容的 Web 地图服务。有关详细信息，请参阅第 331 页第 20 章的 *Web 地图服务* 。

新的示例数据

用于 4.7 版的示例数据在改进之后包括了新的图形示例。其中包括饼图和条形图。有关详细信息，请参阅 *MapXtreme Java 示例数据说明*文档。

日志记录

MapXtreme Java 可以记录多种消息类型。有关详细信息，请参阅第 389 页附录 E 的*系统日志记录* 。

将 MapXtreme Java 4.5 应用程序移植到 4.7

将 4.5 应用程序移植到 4.7，务必注意 API 中建议不使用的过时项目。有关已经过时的类、字段、方法和构造器的完全列表，请参阅 HTML API 文档。

过时的 Layer 类

最主要的过时项目在于 Layer 类。 MapXtreme Java 现在提供 FeatureLayer 类来替代 Layer。FeatureLayer 表示地图的数据图层。例如，表示全球国界区域对象的图层即为 FeatureLayer。

FeatureLayer 中的方法可用于设置图层的可见性和外观。例如，它允许缩放图层。有时，可能只需在特定的缩放级别显示图层。缩放图层只允许在地图的缩放级别处于预设距离之内时显示图层。

例如，假设有一个街道图层和一个邮政编码边界图层。当缩小超过 10 公里或近似距离时，街道的外观将会消失。这是因为缩放 （窗口宽度）太宽，无法显示街道地图的细节。借助于缩放图层，MapXtreme 即可只在缩放设置为允许正确看到街道细节的距离的情况下显示街道图层，例如设置为小于 4 公里的距离。目前还可为各个图层设置不同的缩放图层级别。

图层的搜索方法对于由图层引用的数据依然有效。当显示在地图中时，所有其他方法均和图层外观有关。在某些情况下，对于一个图层而言，表的关系可以为多对一。

此类没有公共构造器。它是在调用 Layers.add 方法时创建的。

重现编译现有应用程序

尽管可能并非必要，但是我们建议重新编译现有应用程序，并使用 -deprecation 标记。如果已经实施定制的 DataProvider，则必须重现编译现有应用程序。

第 1 部分：使用 MapXtreme Java

第 1 部分介绍 MapXtreme Java 以及其为辅助 Web 应用程序开发所提供的众多特性和便利。

主题：

- **第 2 章：入门**

  包括产品内容、系统要求、安装和设置指导。
- **第 3 章：应用程序规划**

  在安装之后，下一个决策是要构建哪类 Web 应用程序。本章定义配置和部署选项的供选范围。
- **第 4 章：地图绘制概念**

  本章介绍构建地图绘制应用程序所需具备的基本地图绘制概念。
- **第 5 章：管理 MapXtreme Java**

  本章介绍管理工具 MapXtreme Java 管理器，以及其中所含的地图定义管理器、命名资源面板和用于应用程序原型开发的 Web 应用程序构建器。
- **第 6 章：Web 应用程序构建器**

  本章还详细介绍了原型向导，并进一步分析了有关 Java 服务器页标记这一原型构建器基础的使用。
- **第 7 章：MapXtreme JavaBeans**

  本章介绍 MapXtreme JavaBeans，这一预置 Java 组件可用于在应用程序中提供地图绘制功能，无需编写代码。
- **第 8 章：编写定制 Servlet**

  本章面向需要编写调用 MapXtremeServlet 的 servlet 以进一步扩展 Web 服务器功能的开发人员。

2 入门

本章介绍 MapXtreme Java 版中包括的组件以及其安装和初始化的方法。

本章内容：

系统要求

MapXtreme Java 版可用于在支持 Java 虚拟机的任何平台上开发地图绘制应用程序。以下是执行地图绘制应用程序的最低要求：

- 支持 servlet/Java 服务器页的 web 服务器，带有支持 servlet/JSP 的插件的 web 服务器，或独立的 servlet 容器。Servlet 容器或插件必须支持 2.3 Servlet API 规格和 1.1 JSP API 规格。
- 与 Java 2 平台兼容的虚拟机 1.4.1 或更高版本。
- 安装在服务器上的视频卡。在 Solaris 上，或者为视频卡，或者为 X11 服务器 （根据 Java 2D 增强图形功能的要求）。
- 200 MB 硬盘空间，用于 MapXtreme Java （~760 MB 用于安装）。
- 400 MB 硬盘空间，用于存放示例地图数据。
- 256 MB RAM，供 MapXtreme 使用。

安装

MapXtreme Java 可安装在支持 Java 2 虚拟机 （1.4.1 或更高版本）的任何系统上。一般过程如下：

1. 安装 MapXtreme Java：安装程序安装 MapXtreme 服务器、客户端、示例、文档和示例部署环境 (Tomcat)。
2. 安装基础地图数据：这是一个单独的安装，将安装多达 400 MB 的地理和人口统计学数据以便在 MapXtreme Java 中使用。通过选择完全安装或定制安装，选择需要哪个数据集。有关说明，请参阅第 23 页的*基础地图数据安装*。

MapXtreme Java 安装

安装 MapXtreme Java：

1. 将 MapXtreme Java 软件光盘放入 CD-ROM 驱动器。
2. 从光盘根目录下运行 **install.htm**。选择所用平台链接：Windows、Solaris、Linux 或 HP-UX。

   或者，对于适当的平台：

   - **Windows -** 在光盘根目录下，转到 \InstData\Windows 并运行 install.exe。
   - **Solaris -** 在光盘根目录下，转到 /InstData/Solaris 并运行 install.bin。
   - **Linux -** 在光盘根目录下，转到 /InstData/Linux 并运行 install.bin。
   - **HP-UX -** 在光盘根目录下，转到 /InstData/Solaris 并运行 install.bin。

3. 在主 MapXtreme 对话框中，选择安装语言 （中文简体）并单击 **OK**。



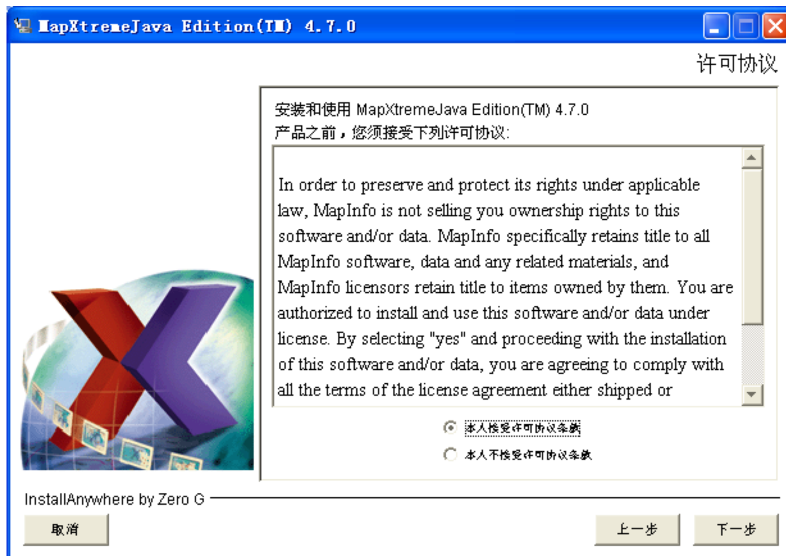
4. 在 “简介”对话框中，阅读面板中的信息，并单击下一步以继续。



5. 在“用户须知”对话框中，复查信息并单击下一步。



6. 选择“本人接受许可协议条款”。单击下一步。显示“选择安装集”对话框。



7. 在 “选择安装集” 对话框中，选择完全安装或定制。单击下一步。完全安装将安装所有产品组件，而定制安装则可选择安装哪些组件。如果选择完全安装，则继续执行第 9 步。

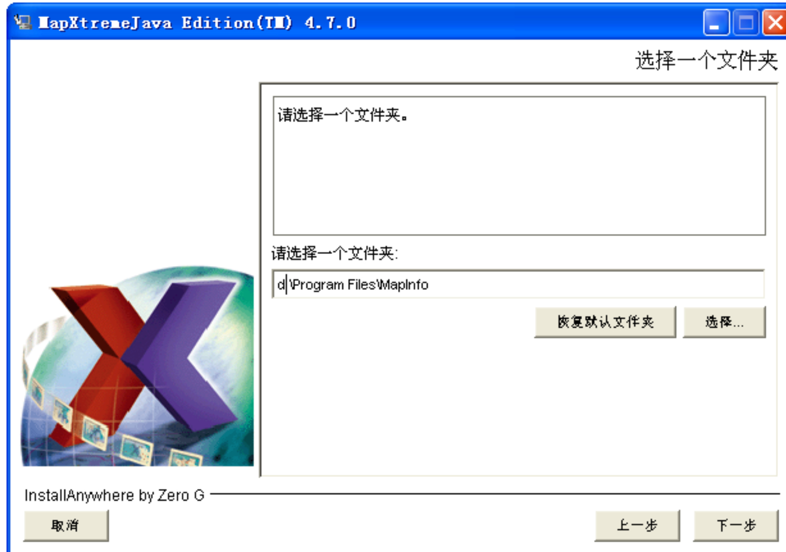


8. 如果在第 7 步中选择 “定制”，则显示 “选择安装组件” 对话框。选择要安装的组件。选项有：
   - 服务器 - 安装服务器组件。
   - 客户端 - 安装客户端组件。
   - 范例 - 安装产品示例。
   - 文档 - 安装产品文档。
   - 部署环境范例 - 安装示例 web 部署环境。这包括 Tomcat 4.1.18。

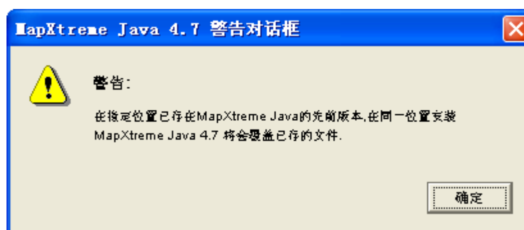
   根据所选组件的不同，以下步骤可能与具体安装过程不相符。单击下一步。显示 “选择安装文件夹” 对话框。



9. 在 “选择一个文件夹”对话框中，接受默认位置安装 MapXtreme Java，或浏览到另一个位置。Windows 默认位置为 <SYSTEM_DRIVE>\Program Files\MapInfo\MapXtreme-4.7.0。在 Solaris、 Linux 和 HP-UX 上，默认位置为 /user/home/mapinfo/MapXtreme-4.7.0。单击下一步。



注：　如果尝试使用的安装位置已经安装了某一版本的 MapXtreme Java，则将收到以下警告：



10. 在 “选择捷径位置”对话框中，接受默认位置安装应用程序图标或浏览到另一个位置，或者选择其他某个可用选项。单击下一步。



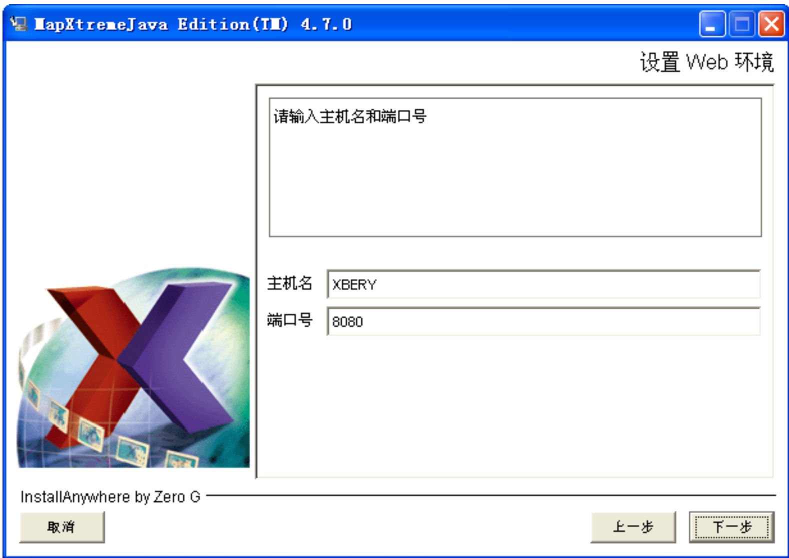
11. 在“选择 Java 虚拟机”对话框中，从可用列表选择一个 Java VM。这必须是 **Java VM 1.4.1** 或更高版本。单击下一步。



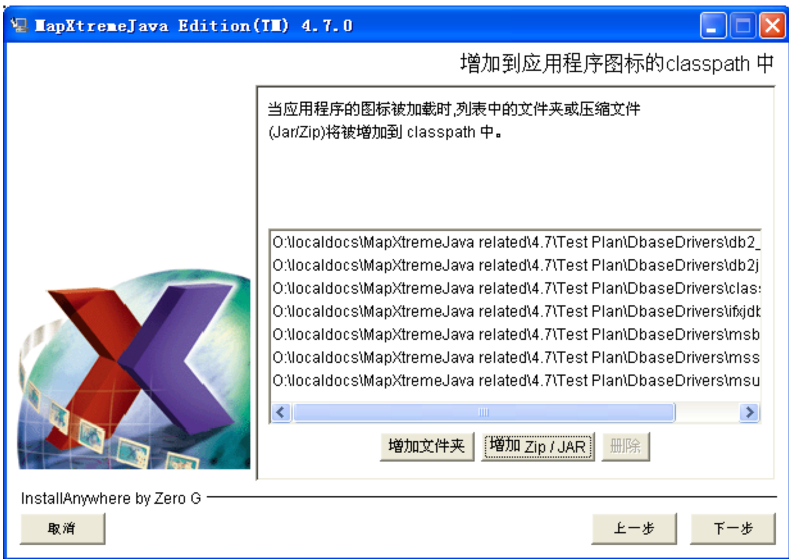
如果尝试选择低于 1.4.1 的 JVM，则会显示以下对话框。



12. 在 “设置 Web 环境” 对话框中，输入主机名和端口号。默认端口为 8080。单击下一步。



13. 在 “增加到应用程序图标的 classpath 中” 对话框中，单击增加文件夹或增加 **ZIP/JAR** 按钮，以添加想要在启动 MapXtreme Java 管理器时出现在 classpath 上的文件，例如包括访问远程数据所需的任何 JDBC 驱动程序。Oracle 用户必须包括 classes12.zip。单击下一步。



14. 显示 “预安装摘要” 对话框。复查信息并单击安装，或单击上一步进行更改。



15. 启动 “正在安装 MapXtreme Java Edition （TM） 4.7.0” 对话框。



16. 在 “安装完毕” 对话框中，单击完成退出软件安装程序。



卸载 MapXtreme Java

要从 Windows 中卸载 MapXtreme Java，在 MapXtreme Java 下的 “程序” 菜单中选择卸载快捷方式，或从目录 <INSTALL_DIR>\Uninstall MapXtremeJava 4.7 下运行 uninstall.exe。

要从 Solaris、 Linux 或 HP-UX 卸载 MapXtreme Java，从目录 <INSTALL_DIR>/UninstallMapXtremeJava-4.7.0 下运行 uninstall.sh。

基础地图数据安装

您可以安装五个地区 （北美洲、南美洲、欧洲、亚洲和澳洲）的全部或部分数据、StreetPro 街道数据和美国人口统计学数据。在安装 MapXtreme Java 时， world.mdf 会自动安装到系统上。

MapXtreme Java 版示例数据说明 PDF 文件介绍了数据集，该文件在安装数据之后位于 <INSTALL_DIR>\Program Files\MapInfo\maps 目录中。

全部数据集大约需要 400 MB 磁盘空间。

安装地图数据：

1. 从 maps 目录中运行 **install.htm**。选择所用平台链接：Windows、Solaris、Linux 或 HP-UX。
   或者，对于适当的平台：
   - **Windows -** 在光盘根目录下，转到 \maps\InstData\Windows 并运行 sampdata.exe。
   - **Solaris -** 在光盘根目录下，转到 /maps/InstData/Solaris 并运行 sampdata.bin。
   - **Linux -** 在光盘根目录下，转到 /maps/InstData/Linux 并运行 sampdata.bin。
   - **HP-UX -** 在光盘根目录下，转到 /maps/InstData/Solaris 并运行 sampdata.bin。
2. 在最初的对话框中，选择适当的语言并单击 **OK**。



3. 阅读 “简介” 对话框并单击下一步以继续。



4. 在 "选择安装文件夹" 对话框中，接受默认位置安装 MapXtreme Java，或浏览到另一个位置。Windows 默认位置为 <INSTALL_DIR>\Program Files\MapInfo\maps。在 Solaris、 Linux 和 HP-UX 上，默认位置为 /user-home/mapinfo/maps。单击下一步。



5. 在 “选择安装集”对话框中，选择典型安装以安装整个基础地图数据集，或选择定制选择要安装的部分数据。单击下一步。如果选择典型安装，则继续执行第 7 步。



6. 如果在第 7 步中选择了定制，则显示 “选择产品组件”对话框。选项有：
   - 北美 - 北美洲的示例地图数据。这包括美国、墨西哥和加拿大的数据。
   - 南美洲 - 南美洲的示例地图数据。这包括巴西和阿根廷的数据。
   - 欧洲 - 欧洲的示例地图数据。这包括法国、英国、葡萄牙、西班牙、德国和意大利的数据。
   - 亚洲 - 亚洲的示例地图数据。这包括日本、以色列、韩国和中国的数据。
   - 澳大利亚 - 澳洲的示例地图数据。
   - 人口统计学 - 人口统计学数据。
   - **STREETPRO** - StreetPro 数据。

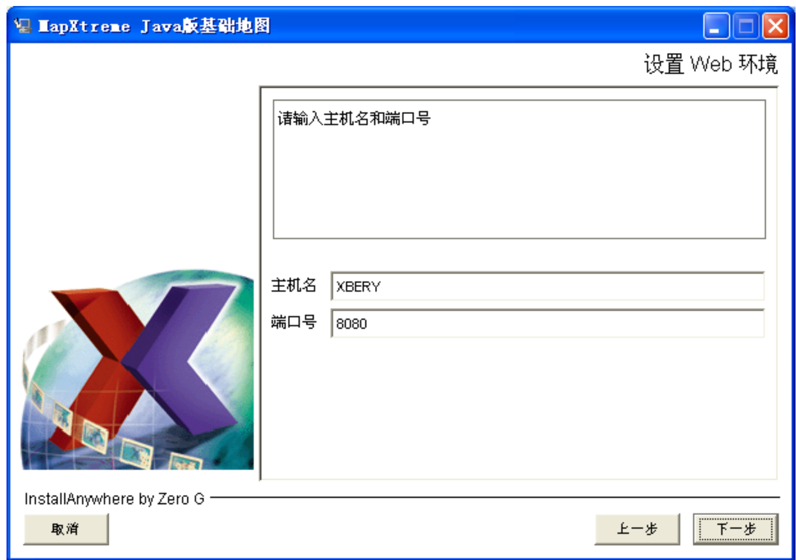
   清除不想安装的数据集的复选框，并单击下一步。



7. 在 “选择捷径位置” 对话框中，接受默认位置安装应用程序图标或浏览到另一个位置，或者选择其他某个可用选项。单击下一步。



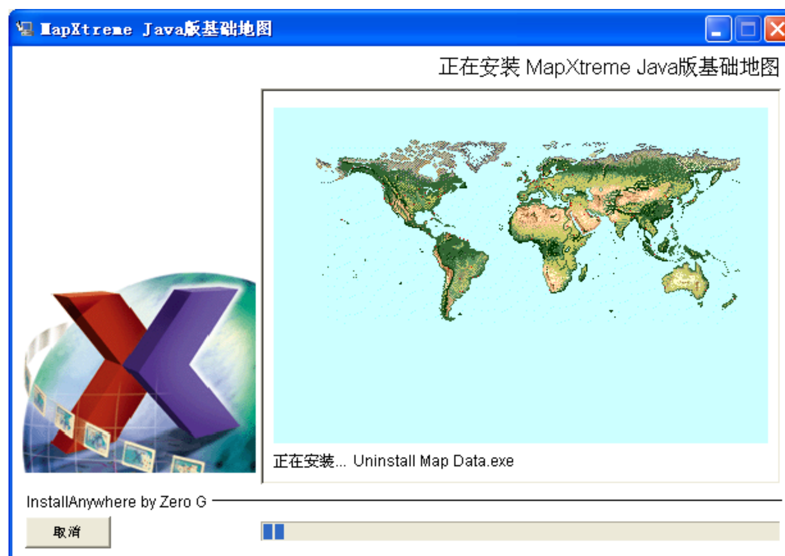
8. 在“设置 Web 环境”对话框中，输入主机名和端口号。默认端口为 8080。单击下一步。



9. 在“预安装摘要”对话框中，复查信息并单击下一步以继续，或单击上一步返回到前一个对话框进行更改。



10. 单击安装。安装过程执行完毕。



11. 单击完成离开数据安装程序。



卸载基础地图数据

要从 Windows 中卸载 MapXtreme Java，在 MapXtreme Java 版 “评估基础地图”下的“程序”菜单中选择卸载快捷方式，或从目录 <INSTALL_DIR>\UnInstallerData 下运行 uninstall.exe。

要从 Solaris、 Linux 或 HP-UX 卸载 MapXtreme Java，从目录 <INSTALL_DIR>/UnInstallerData 下运行 uninstall.sh。

Tomcat 部署环境

MapXtreme Java 使用 servlet 模型处理请求。MapXtreme Java 需要一个 web 服务器 /servlet 容器来运行，比如 Apache/Tomcat、 iPlanet、 WebLogic 等。为了方便用户，我们安装了 Tomcat 4.1.18 并与 MapXtreme Java 相集成，作为产品安装的一部分。

在安装后，可以在 <INSTALL_DIR>\MapXtreme-4.7.0\Tomcat-4.1 下看到 Tomcat 目录。在 <INSTALL_DIR>\MapXtreme-4.7.0\Tomcat-4.1\webapps 目录中，存放的是 MapXtreme Java (/mapxtreme47)、 Mapviewer JSP 示例
(/mapviewer47)、其他示例应用程序 (/samples47) 和 WMS Server (/wmsserver111) 的 servlet 上下文。

多个 Tomcat 实例

使用多个 Tomcat 实例可提高 MapXtreme Java 的性能。在某些具有四个或更多 CPU 的 Solaris 服务器上，当同时运行三个 Tomcat 实例时， MapXtreme 的性能会得到提高。例如，服务器可能运行 Apache 和三个 Tomcat 实例，而 Tomcat 设置 （例如 mod-jk.conf）可在 Tomcat 实例之间执行负载平衡。

如果在同一台计算机上有多个 Tomcat 实例，则需要修改 Tomcat 配置，使每个实例监听不同的端口号。例如，可能需要有 server.xml 文件的多个副本 （例如 server2.xml、server3.xml），每个副本都指定唯一的端口号和唯一的 workDir 设置，这样才能使多个 Tomcat 实例之间不发生冲突。还需要使用可选的 -f 参数启动每个 Tomcat 实例，以指定所要使用的 xml 文件。例如：

```
tomcat start -f ../conf/server2.xml
```

请注意，负载平衡并不必局限于一台计算机。例如，可以使 Tomcat 运行在几台计算机上，利用 Apache 执行负载平衡。

在其他 Web 服务器环境中部署 MapXtreme

作为安装过程的一部分， web 存档 (war) 文件将被创建并放置在 <INSTALL_DIR>\MapXtreme-4.7.0\wars 目录中。war 文件有：

- mapxtreme47.war - 主 mapxtreme 上下文
- mapviewer47.war - JSP 示例 MapViewer web 应用程序
- samples47.war - servlet 示例应用程序
- wmsserver111.war - Web 地图服务 (WMS) 的上下文

这些 war 文件可部署到支持 WAR 文件的任何 servlet 容器中。

注： 这些文件中的信息将它们与其所在的主机联系起来。

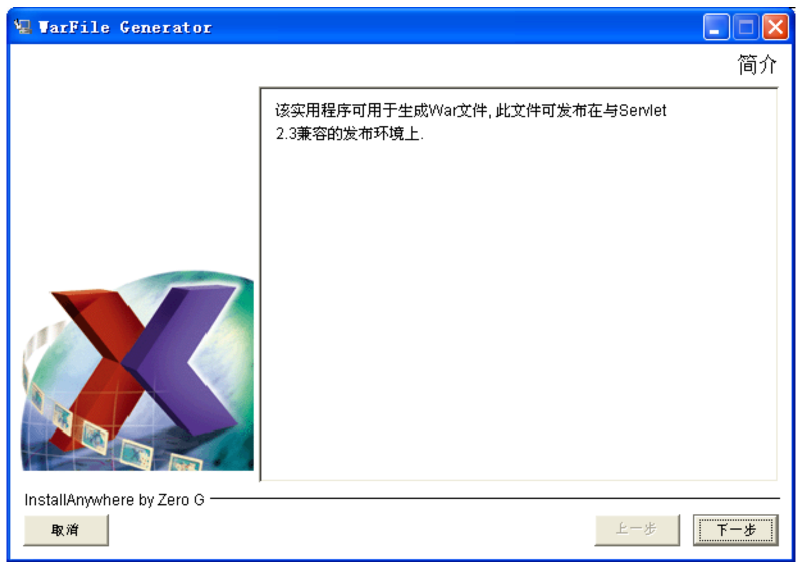
使用 WARFile 生成器

运行 WARFile 生成器：

1. 从 war-utility 目录中运行 **install.htm**。选择所用平台链接：Windows、 Solaris、Linux 或 HP-UX。
   或者，对于适当的平台：
   - **Windows -** 在光盘根目录下，转到 \war-utility\InstData\Windows 并运行 install.exe。
   - **Solaris -** 在光盘根目录下，转到 /war-utility/InstData/Solaris 并运行 install.bin。
   - **Linux -** 在光盘根目录下，转到 /war-utility/InstData/Linux 并运行 install.bin。
   - **HP-UX -** 在光盘根目录下，转到 /war-utility/InstData/Solaris 并运行 install.bin。
2. 在最初的对话框中，选择适当的语言并单击 **OK**。



3. 在 “简介” 对话框中，阅读面板中的信息，并单击下一步以继续。



4. 在 “选择安装集” 对话框中，选择默认 **MAPXTREME WAR** 文件选项以创建表示 MapXtreme servlet 上下文和 MapXtreme 示例上下文的 war 文件，或者选择用户指定的上下文。该选项可用于选择指向 servlet 上下文的文件夹，并用指定名称在指定位置创建一个 war 文件。单击下一步。



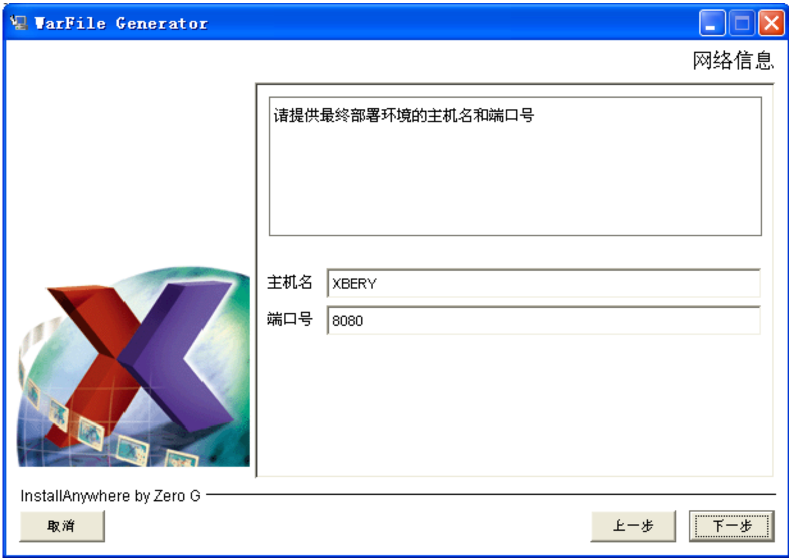
5. 在 “选择安装文件夹” 对话框中，如果在上一步骤中选择了默认 **MAPXTREME WAR** 文件，则选择一个目录以安装表示该上下文文件的 war 文件。单击下一步。

如果在上一步骤中选择了用户指定的上下文，则选择要创建 war 文件的目录。单击下一步。

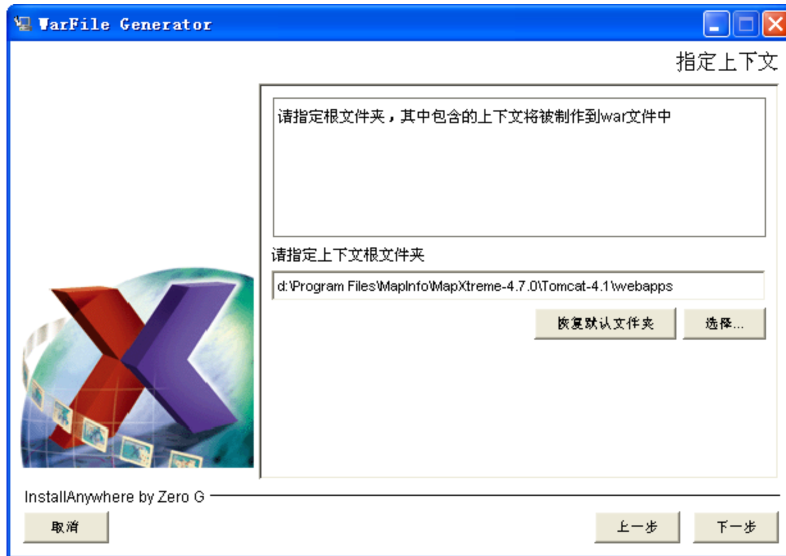


6. 在 “网络信息”对话框中，输入主机名和端口号。单击下一步。

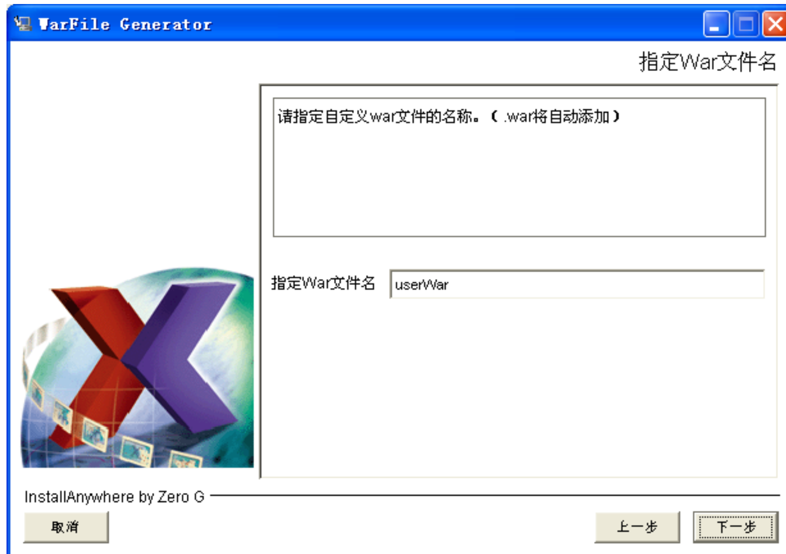
如果在第 4 步中选择了默认 **MapXtreme War** 文件，则单击下一步继续执行第 9 步。



7. 在“指定上下文”对话框中，指定包含要放入 WAR 文件的上下文的目录。单击下一步。



8. 在“指定 War 文件名”对话框中，指定要赋给定制 WAR 文件的名称。单击下一步。



9. 在 “增加到应用程序图标的 classpath 中” 对话框中，单击增加文件夹或增加 **ZIP/JAR** 按钮，以添加想要在启动 MapXtreme Java 管理器时出现在 classpath 上的文件。例如，包括访问远程数据所需的任何 JDBC 驱动程序。Oracle 用户必须包括 classes12.zip。单击下一步。



10. 在 “MapXtreme 记录文件的位置” 对话框中，指定生成日志文件的位置。有关日志记录的详细信息，请参阅第 389 页附录 E 的*系统日志记录*。单击下一步。



11. 显示 “预安装摘要”对话框。复查信息并单击安装，或单击上一步进行更改。



12. 显示 “正在安装 WarFile Generator”对话框。



13. 在 “安装完毕” 对话框中，单击完成离开安装程序。



已安装的组件

下面概要介绍 MapXtreme Java 中包括的主要组件。每个组件后面有一个简短说明。

MapXtreme Java 管理器

这是一个 GUI 工具，可用来管理 MapXtreme Java 的若干方面，包括创建和加载地图定义、管理命名资源、管理 JDBC 与 RDBMS 的连接、以及在向导的帮助下构建原型和应用程序。MapXtreme Java 管理器将在*第 5 章：管理 MapXtreme Java* 中加以讨论。

MapJ API 和 Java 类库

MapJ API、类库和 Javadocs 提供了构建自定义的应用程序、 applet、 servlet 和 JavaBeans 所需的所有内容。开发人员指南中的第二部分讨论了 MapJ API。

JSP 标记库

MapXtreme Java 包括一组定制 JSP 标记，可帮助您快速构建原型和应用程序。有关详细信息，请参阅*第 6 章：Web 应用程序构建器* 。

定制符号

MapXtreme Java 安装一个定制符号 GIF 图像的文件夹。关于每个图像的缩略图，请参阅*附录 F: 定制符号* 。有关在应用程序中使用这些符号的信息，请参阅*第 14 章：标注和样式* 。

MapXtreme Java 还提供了画笔、画刷和符号字体的预定义样式。使用这些样式可自定义地图图层。相应样式位于 /resources/mistyles 下的 mapxtreme47 servlet 上下文中。

示例

MapXtreme Java 包括各种示例 servlet 和 applet，展示了 MapXtreme Java 的关键组件。其中大多数在安装后位于 /examples 目录中。这些示例应用程序中的若干程序包括说明其特性和代码的 HTML 教程。此外，也可以在 MapXtreme-4.7.0/lib/client/mxjclientsamples.jar 和 MapXtreme-4.7.0/lib/server/mxjserversamples.jar 中找到与编译的版本。

安装字体

MapXtreme Java 提供了 11 个 TrueType 字体集，用作地图符号。这些字体安装在 Windows 上并自动注册。它们位于 <SYSTEM_DRIVE>/<OS>/fonts 目录中。

在 Windows 以外的平台上，安装程序将把这些字体复制到 <SYSTEM_DRIVE>/<OS>/fonts 目录中。安装之后，必须在操作系统中注册这些字体，然后才能使用它们。下面是针对 Solaris 的说明。对其他平台的说明与此相似。此说明假定您已成功地安装了 Java 2 和 MapXtreme Java。

1. 以 ROOT （即超级用户）身份登录 Solaris。
2. 执行 Solaris 命令 `xset -q`，该命令可显示系统用于字体存储 / 使用的所有目录。
3. 将 MapXtreme Java 字体复制到其中某个目录中 （即 /usr/openwin/lib/X11/fonts/TrueType）。
4. 通过执行以下命令，重新指定对这些字体文件的权限：
```
chmod 644 Map*.ttf
chmod 644 map*.ttf
```
5. 编辑 fonts.dir 文件。转至存放字体的目录，并为每个 MapXtreme Java 字体提供一个条目。可以从 fonts_sample.dir 文件复制这些条目，该文件在安装后位于 mapxtremejava/server/fonts 目录中。这些条目为：
```
mapiau__.ttf -unknown-mapinfo arrows unicode-regular-r-normal--0-0-
0-0-p-0-iso8859-1
mapicuc_.ttf -unknown-mapinfo cartographic unicode-regular-r-
normal--0-0-0-0-p-0-iso8859-1
mapimu__.ttf -unknown-mapinfo miscellaneous unicode-regular-r-
normal--0-0-0-0-p-0-iso8859-1
mapiogu_.ttf -unknown-mapinfo oil&gas unicode-regular-r-normal--0-
0-0-0-p-0-iso8859-1
mapireu_.ttf -unknown-mapinfo real estate unicode-regular-r-normal-
-0-0-0-0-p-0-iso8859-1
mapisu__.ttf -unknown-mapinfo shields unicode-regular-r-normal--0-
0-0-0-p-0-iso8859-1
mapitu__.ttf -unknown-mapinfo transportation unicode-regular-r-
normal--0-0-0-0-p-0-iso8859-1
mapiwu__.ttf -unknown-mapinfo weather unicode-regular-r-normal--0-
0-0-0-p-0-iso8859-1
mapsu___.ttf -unknown-mapinfo symbols unicode-regular-r-normal--0-
0-0-0-p-0-iso8859-1
mapsymu.ttf -unknown-map symbols unicode-regular-r-normal--0-0-0-0-
p-0-iso8859-1
mapispu___.ttf -unknown-map symbols unicode-regular-r-normal--0-0-
0-0-p-0-iso8859-1
```
6. 在 fonts.dir 文件的第一行中，对数字加 10 可包括要增加的 10 个新字体条目。
7. 退出 Solaris，并以任意用户身份再次登录。

8. 执行 Solaris 命令 `xlsfonts`，该命令将读取所有现有的 fonts.dir 文件并显示系统可使用的所有字体列表。
9. 在启动 X 服务器之前启动字体服务器 （如果您正在使用的话）。

注： 没有 ROOT 访问权的用户也可以通过以下方法使 MapXtreme 字体生效。

假定字体位于 /data/my_data/MXTJ47/fonts 中，将环境变量 JAVA_FONTS 设置为：

```
/usr/openwin/lib/X11/fonts/TrueType:/data/my_data/MXTJ47/fonts
```

在没有图形卡的 UNIX 上安装

虽然可以手动安装 MapXtreme Java，但 MapInfo 强烈建议在使用了图形卡的计算机上安装。如果必须执行手动安装，请使用 Xwindows 选项或 “控制台”选项，如以下各节所述。

X-windows 安装选项

在其 UNIX 系统中没有图形卡的 UNIX 开发人员需要使用 X-windows (X11) 来运行 MapXtreme Java 安装程序。使用 X 仿真程序 （如 eXceed 或 KEA!X）时，可以在远程系统上安装，而不必在计算机本地安装。可以在 X 中安装产品，但会出现问题。通常尽量确保使用相应软件的验证版本。如果设置不正确，显示管理器可能会出现问题。由于 MapInfo 不生成这些 X 管理器，因此我们建议您在 GUI 模式下运行安装程序，然后使用安装了图形卡的 UNIX 系统。执行以上操作之后，按照第 14 页的*安装*一节所描述的 Solaris 安装步骤进行操作。

控制台安装选项

如果在没有图形卡 （以支持 GUI 安装）的 UNIX 计算机上进行安装，并且没有 X-windows，则可以执行控制台安装并手动配置文件。可以使用 MapXtreme Java，但不能运行随 MapXtreme Java 提供的示例。

要在控制台模式下从命令行运行安装程序，请键入命令：

```
install.bin -i console
```

在执行 MapXtreme Java 的控制台安装时，产品在默认情况下安装在 /<USER_HOME>/MapXtremeJava-4.7.0。下表说明在执行控制台安装之后需要对 web.xml 文件进行的更改。

注： 在控制台模式下启动 MapXtreme Java 安装程序时，系统将提示您选择要运行安装程序的地区。选择指向所在地区的编号。

位置：安装产品的地方。

在 UNIX 平台上，默认安装位置为 /<USER_HOME>/MapXtremeJava-4.7.0。

文件：需要进行配置以便使用 MapXtreme Java 的文件。

原始和替换文本：MapXtreme Java 安装程序会配置一些文件，以便在计算机上部署产品。它将用具体环境的特定值来替换这些文件中的文本，从而实现配置。在命令行提示符下运行安装程序时，不执行此配置。因此，需要在每个文件中手动替换文本。需要更改的文件以及在每个文件中需要替换的文本如下：

位置和文件	原始文本	替换文本
<USER_INSTALL_DIR> /MapXtreme-4.7.0/ binMapXtreme Java 管理器 Web Server.lax	lax.command.line.args= http://:/mapxtreme47/manager	lax.command.line.args= http://@hostname@:@port.number@ /mapxtreme47/manager
<USER_INSTALL_DIR> /MapXtreme-4.7.0/Tomcat-4.1/conf/server.xml	<Connector className="org.apache.coyote.tomcat4.CoyoteConnector" port="" minProcessors="5" maxProcessors="75" enableLookups="true" redirectPort="8443" acceptCount="100" debug="0" connectionTimeout="20000" useURIValidationHack="false" disableUploadTimeout="true" />	<Connector className="org.apache.coyote.tomcat4.CoyoteConnector" port="@port@" minProcessors="5" maxProcessors="75" enableLookups="true" redirectPort="8443" acceptCount="100" debug="0" connectionTimeout="20000" useURIValidationHack="false" disableUploadTimeout="true" />

位置和文件	原始文本	替换文本
<USER_INSTALL_DIR> /MapXtreme-4.7.0/Tomcat-4.1/webapps/ wmsserver111/WEB-INF/ web.xml	<init-param> <param-name>providerURL </param-name> <param-value>http://:/mapxtreme47/ resources </param-value> </init-param>	<init-param> <param-name>providerURL </param-name> <param-value>http:// @hostname@:@ port.number @/ mapxtreme47/resources </param-value> </init-param>
USER_INSTALL_DIR>/ MapXtreme-4.7.0/docs/ index_offline.html	online version </p>	online version</p>

配置考虑因素

在执行控制台安装之前应考虑以下几项。

- 基于 MapXtreme Java 控制台的安装不配置运行示例所需的文件，因此在运行安装之后，任何示例均无法工作。若要配置，必须修改每个示例的特定文件。
- 若要在 MapXtreme Java 管理器 applet 中加载示例数据，必须用您的计算机名来重命名安装根目录下的 examples 目录中的某一目录。例如，更改

  <USER_INSTALL_DIR>/MapXtreme-4.7.0/examples/server/data/machine-name

  为

  <USER_INSTALL_DIR>/MapXtreme-4.7.0/examples/server/data/@hostname@

  其中 hostname 为您的计算机名。

配置故障排除

在安装 MapXtreme Java 并在一个 servlet 容器中进行设置之后，如果遇到配置问题而令 MapXtremeServlet 无法处理地图绘制请求，可使用以下信息。例如，如果尝试部署示例 applet 一 SimpleMap，该 applet 可能会报告一个 FileNotFoundException，指出 MapXtremeServlet 的 URL 有问题。

如果遇到了这样的问题，那么主要可能有两个原因：

- MapXtremeServlet 的配置可能不正确。
- 可能未指定正确的 URL 来访问 MapXtremeServlet。

要解决此类故障，应确保服务器已启动，并运行 “调试” URL。此时将显示一个状态页，指示 MapXtremeServlet 正在运行且 URL 正确。在浏览器的 URL 文本字段中，键入指向 MapXtremeServlet 的 URL，后面跟 “?debug=true”。例如：

```
http://hostname:portnumber/mapxtreme47/mapxtreme?debug=true
```

状态页将显示基本的诊断信息，包括 Java VM 的版本和使用中的 MapXtreme 的版本。如果看不到此状态页，则要么没有运行 servlet，要么 URL 不正确。

验证 MapXtremeServlet 是否运行

除了运行调试 URL 的方法之外，还可通过查看开始过程来检查 MapXtremeServlet 是否运行。当 servlet 容器启动时，查看是否有任何错误迹象。例如，如果 servlet 容器具有可查看的控制台窗口或日志，则查看控制台或日志，检查是否有任何消息包含 “MapXtremeServlet” 或 “mapxtreme”。

如果 servlet 容器提供了管理员工具，则可通过运行管理员来检查 MapXtremeServlet 的状态。如果 servlet 容器没有提供管理员工具，则尝试查看控制台窗口或日志来检查 MapXtremeServlet 是否启动。

显示在控制台窗口或日志中的具体文本部分取决于您使用了哪个 servlet 容器。一般来说，如果 MapXtremeServlet 初始化成功，即会看到初始化消息。例如，如果已使用 Tomcat 正确配置了 MapXtreme，则 Tomcat 控制台或日志文件会显示以下文本：

```
Context log path="/mapxtreme47" :mapxtreme: init
```

这些消息表示已处理了 MapXtreme 的 web.xml 文件，并正确初始化了 MapXtremeServlet。请注意，上面的消息包含了 “mapxtreme: init” 而不是 “com.mapinfo.mapxtreme.MapXtremeServlet: init”。这是因为 MapXtreme 的安装程序为 MapXtremeServlet 分配了一个注册名称 “mapxtreme”；请参阅 /mapxtreme47/WEB-INF 中的 MapXtreme web.xml 文件。

如果当启动 servlet 容器时 MapXtremeServlet 没有初始化，这并不一定表示存在问题。它只是表示 MapXtremeServlet 配置没有将 “Load On Startup” 选项设置为 “True”（因此，未初始化 MapXtremeServlet 是因为还没有对此发出请求）。MapXtreme 的 Tomcat 安装程序不会将 “Load On Startup” 设置为 “True”，因此当启动 Tomcat 时， Tomcat 用户应当看到如上所述的初始化消息。

如果在验证 MapXtreme 配置是否正确时有困难，则应配置 MapXtreme，使其设置为 “Load On Startup”，然后重新启动 servlet 容器。在 Tomcat 中，“Load On Startup” 选项是在 mapxtreme/WEB-INF/web.xml 中使用 <load-on-startup> 标记指定的。如果 servlet 容器提供了管理员工具，则可以在管理员中设置 “Load On Startup” 选项。

如果 MapXtremeServlet 部署说明符设置为 “Load On Startup”，但您仍然没有在 servlet 容器的控制器或日志中看到 MapXtremeServlet 初始化消息，那么可能是 MapXtreme 的配置方式出现问题；在这种情况下，复查配置 MapXtreme 的步骤，确保所有配置步骤均已正确执行。

验证 MapXtremeServlet 的 URL 是否正确

在利用 MapXtremeServlet 之前，需要知道其准确的 URL。例如，在运行示例 applet －SimpleMap 之前，需要编辑加载 applet 的 HTML 文件，并指定 MapXtremeServlet 的 URL。

注：　URL 区分大小写。

MapXtremeServlet URL 的一般形式如下：

```
http://hostname:portnumber/path/registeredname
```

hostname：这取决于服务器的名称或 IP 地址。如果在一台计算机上执行所有任务 （例如，测试或开发），则可以使用 “localhost” 作为主机名。

portnumber：这取决于您如何设置 servlet 容器。如果运行了 MapXtreme 的安装程序并接受了默认设置，这应该是端口 8080，当指定 MapXtreme URL 时可以忽略端口号 （和冒号）。

path：这取决于 servlet 容器，还可能取决于创建了哪些子目录。请参阅下面的示例。

registeredname：这是一个可以指定的名称，例如 “mapxtreme”，它是 com.mapinfo.mapxtreme.MapXtremeServlet 类的快捷方式。并非严格要求使用快捷方式；此时可以指定 com.mapinfo.mapxtreme.MapXtremeServlet 作为 URL 的最后一部分。但是，如果已设置了 MapXtremeServlet 的注册名称，则应使用此注册名称。 （您在部署说明符中指定的任何配置选项 （比如 “Load On Startup” 选项）与此注册名称相关联；如果使用指定类名称而不是注册名称的 URL，那么这些选项将不起作用。）

示例：Tomcat

如果在使用随 MapXtreme Java 提供的示例部署环境 (Tomcat)，并且接受了默认设置，那么 MapXtremeServlet URL 应当是：

```
http://stockholm:8080/mapxtreme47/mapxtreme
```

在此示例中， URL 的 /mapxtreme47 分段表示 mapxtreme 目录创建在 Tomcat 目录中。结束分段 /mapxtreme 表示安装程序使用 <servlet-name> 标记放置在 [tomcat directory]/mapxtreme47/WEB-INF/web.xml 中的注册名称。

LAX 文件

MapXtreme Java 安装程序在系统中安装若干配置文件，这些文件与 MapXtreme Java 管理器的可执行文件和卸载程序相关联。这些文件的扩展名为 .lax （如 MapXtreme Java 管理器 .lax），提供了有关系统的重要配置信息：

- lax.class.path - 将所需的任何 JDBC 驱动程序添加到此处的 jar 文件列表中。例如，要访问有空间选项的 Oracle，添加 classes12.zip 至 classpath。安装程序提供了添加任何 .zip 或 .jar 文件的机会。不过，如果需要其他文件，可以在安装后在此处进行添加。
- lax.stderr.redirect - 在调试时，将 “=console” 放在最后以便发送信息至控制台窗口。
- lax.stdout.redirect - 在调试时将 “=console” 放在最后，同上。
- lax.nl.current.vm - 如果 VM 不工作，则检查其版本。

软件复制保护

如果您的 MapXtreme Java 版副本受到复制保护 （即在 MapXtreme Java 管理器中显示的地图上可见水印 "Protected Software - Key Required for Permanent Use"），则需要申请许可文件，以便去除水印，使 MapXtreme Java 能够长期使用。

从地图显示中去除水印：

1. 执行以下操作之一：

   填写 MapXtreme Java 产品包内提供的名为 “重要须知” (Important Please Read) 的复制保护密钥申请。

   从 MapXtreme 网站下载申请表，网址为：**http://testdrive.mapinfo.com/mapxtremejava**。
   单击有链接标记的 Support，然后单击 Documentation 链接，选择 “Software Copy Protection License Application”。
2. 将完成后的表格提交给经销商或分销商。查看表格以获得联系信息。经销商会处理相应申请，并将许可文件通过电子邮件发送给您。
3. 收到名为 mapxtremejava.key 的许可文件之后，将其放在 classpath 上的目录中，例如 C:/Program Files/MapInfo/MapXtreme-4.7.0/lib/client。此文件还需要放在服务器的 classpath 上，因此对于 MapXtremeServlet 来说，也要放在 mapxtreme47/WEB-INF/classes 目录中。如果还没有 /classes 目录，则需创建该目录。此时地图即可正常显示，不带水印。

3

应用程序规划

本章介绍基于 Web 部署基础架构的要求和必要技巧，同时还提供了 MapXtreme Java 组件和公共配置概览。

本章内容：

Web 部署

基于 Web 的部署提供了众多优点，其中有：

- 降低所有者成本：通过 Web 分发应用程序的成本比较经济，因为消除了为访问应用程序而在每台计算机上安装组件的需求。
- 可扩展性：Web 系统通过功能强大的服务器提供服务。如果用户数量增加，那么可以随之添加更多的服务器。
- 数据和软件的访问：应用程序的每个用户均可访问最新的版本。对于所有用户而言，软件和数据更新只需针对服务器操作即可。
- 安全：由于对安全采用了集中式的管理，因此对于具体实施方案可以具有更多控制。

一般而言，基于 Web 部署的工作如下所示。在客户端，用户通过 HTML 页面和 / 或其浏览器中的 applet 与地图绘制应用程序交互。交互是以请求 / 响应方案为基础的。例如，用户发出请求，要求放大地图上的某一区域。该请求将传送到 Web 服务器。服务器端的应用程序将与地图服务器通信，提供请求的信息。更新的地图将返回至嵌入到 HTML 页面或 applet 中的客户端浏览器。创建地图所需的任意数据都将驻留在 RDBMS 之中，数据库调用经由 JDBC 检索数据完成。有关基于 Web 部署的更多详细说明，请参阅第 133 页第 8 章的*编写定制 Servlet* 。

无线通信业就是基于 Web 的应用服务器的一个实例。潜在的客户经常需要知晓其位置是否位于通信公司的覆盖范围之内。通过提供嵌入到 HTML 页面中的覆盖区域地图并经由因特网部署，电讯公司即为客户采用自助方式提供及时的响应。客户可以通过使用放大或平移等导航工具，对地图进行交互操作。每个点击均相当于一个发送到电讯应用程序用于重新生成地图的请求。

基础架构要求

通常，基于 web 的地图绘制部署利用标准的组件，其中包括运行 web 服务器的服务器、应用服务器 （可以和 Web 服务器是同一服务器）和用于背景地图的地图数据数据库、以及应用程序相关的定制数据。

MapXtreme Java 基于 Web 的部署需要 Java 2 的支持。此外，由于用于 MapXtreme Java 的地图绘制服务器是作为 servlet 部署的，因此 Web 服务器必须支持 servlet。地图数据可在本地存储，也可以经由 JDBC 从 RDBMS 访问。要构建的应用程序可以采用 servlet、 Java 服务器页、 Enterprise JavaBean 或 applet 等多种形式。

必要技巧汇总

MapXtreme Java 是一个面向编写 servlet 和 applet 的 Java 开发人员的产品。在构建应用程序时，有必要了解创建 Java GUI 的技巧。

此外，由于是在 Web 上部署 MapXtreme Java，因此还应具备项目所需的 web 开发和 HTML 专业知识。所需具体内容将取决于所用的开发工具。

如果要访问 RDBMS 中的数据，则还需要数据库使用和管理方面的技巧。

最后，由于正在创建的是地图绘制应用程序，熟悉地图绘制的概念和 / 或 MapInfo 地图绘制产品也会对开发有所助益。有关地图绘制的基本知识，请参阅*第 4 章： 地图绘制概念* 。

部署选项

用于 MapXtreme Java 的部署选项可以划分为以下三类：瘦客户机、中型客户机和胖客户机。其间的区别在于软件和向客户机发送的数据的多少。

各种类型的概览如下所示，后续内容中还提供了构成 MapXtreme Java 的组件的有关讨论。

瘦客户机

在瘦客户机的部署中，用户将与浏览器中的 HTML 页面交互。地图通常是嵌入到 HTML 中的 GIF 图像。地图请求处理在服务器端完成。这是典型的因特网部署，客户机上无需 Java。

要构建此类应用程序，需要了解如何开发服务器端的应用程序以生成 HTML。

胖客户机

胖客户机是备选的另一种方案。客户机下载一个 Java applet，由其提供比直接的 HTML 更加复杂的用户界面。此外， MapXtreme Java 可以返回向量数据来取代栅格数据。由于增加了用于 applet 的下载时间，因此这一方案更加适合于在内网系统部署，以便可以更好地控制客户端。

要构建此类应用程序，需要了解如何构建 Java applet，并可以选择使用 JavaBeans。

中型客户机

中型客户机是介于瘦客户机和胖客户机之间的选项。与胖客户机相同，中型客户机下载 applet，因此客户机必须支持 Java。类似瘦客户机，中型客户机接收地图的栅格图像。applet 可以提供比直接 HTML 更加复杂的用户界面和附加的地图工具，例如选取框工具。

要构建此类应用程序，需要了解如何开发 applet 和服务器端的应用程序交互。

MapXtreme Java 概览

MapXtreme Java 版有 4 个主要组件，如下所示：MapXtremeServlet、 MapJ 对象、数据提供方和渲染器。这些组件协同工作，可用于访问地理数据、控制数据并为应用程序提供地图或数据。

MapXtremeServlet

MapXtremeServlet 是在 MapXtreme Java 产品中提供的地图绘制服务器。该服务器处理 3 类客户机请求：

- 地图图像请求
- 向量地图数据请求
- 地图元数据请求 （如图层的列名）。

MapXtremeServlet 对源自 MapJ 对象的 HTTP POST 请求作出响应。此外，还可使用 MapXtreme Java 的 XML 协议编写定制客户端，用于和 MapXtremeServlet 通信。

MapXtremeServlet 设计利用其父级 servlet 容器的功能。 MapXtremeServlet 为无状态，依赖客户机请求来完全说明地图状态。图像请求在 MapXtremeServlet 之中由多线程的渲染器服务器处理。与此类似，地图数据请求由多线程的 DataProvider 服务器处理。这些因素令 MapXtremeServlet 在父级 servlet 容器中部署时，具有相当高的可扩展性。

在 MapXtremeServlet 侧重于完成地图绘制任务的同时，其父级 servlet 容器处理负载平衡、安全和容错等其他问题。 Servlet 容器可见于类似 Sun 的 JavaWebServer 的 Web 服务器，以及类似 BEA 的 WebLogics 的应用服务器。类似 Apache 的 Web 服务器或 Microsoft IIS 的 Web 服务器不含 servlet 容器。此时必须使用单独的 servlet 容器插件， JRun 或 Tomcat。

MapJ 对象

MapJ 对象管理地图的状态。该对象维护地图中心和缩放、坐标系、距离单位和共同构成地图的图层。 MapJ 位于 MapXtreme 客户机 API 的最顶层。

MapJ 对象可以配置用于与不同类型的渲染器和数据提供方协同工作。在最为典型的配置中， MapJ 是 MapXtremeServlet 的客户机。 MapJ 向 MapXtremeServlet 实例发送请求，请求的一部分为 servlet 提供了其当前状态。 MapJ 从 servlet 获取地图图像和数据。

MapJ 还可以独立工作，直接获取地图数据并生成地图图像。 MapXtreme 允许 MapJ 配置为与其他变更版本协同工作，这正是 MapXtreme 基于组件的设计方案的强大之处。例如， MapJ 可配置为通过一个或多个 MapXtremeServlet 实例访问地图数据，同时仍然可以显示地图图像。

由于 MapJ 的主要目的是维护地图状态，因此其占用的内存较小。 MapJ 正是由此成为在中间层或 n 层体系结构中实施的理想之选。有关部署配置的详细信息，请参阅第 52 页的*设计时的考虑因素* 。

渲染器

渲染器显示地图数据。共有以下 5 种类型的渲染器：LocalRenderer、 MapXtremeImageRenderer、 EncodedImageRenderer、 IntraServletContainerRenderer 和 CompositeRenderer。

LocalRenderer 可从任意 Java Graphics 2D 对象创建，通常从 AWT 组件或 BufferedImage 获取。 LocalRenderer 位于 “本地”或在与其关联的 MapJ 对象的同一处理空间之内。其使用数据提供方来直接获取地图中每个图层的图元。 LocalRenderer 随后将图元绘制到其组件的图形对象中。

MapXtremeImageRenderer 可以从 MapXtremeServlet 实例的 URL 引用创建。当 MapJ 使用 MapXtremeImageRenderer 时，表示其将令 MapXtremeServlet 的实例控制地图绘制。 servlet 通过向 MapJ 对象返回栅格图像来满足这一请求。 MapXtremeServlet 支持包含 GIF、 JPEG 和 PNG 之内的多种栅格格式。此外， MapXtremeServlet 的渲染器服务器还通过使用 LocalRenderer 的实例来满足渲染请求，并将图像导出为所需的栅格格式。

MapXtreme Java 还提供有关 MapXtremeImageRenderer 的各种变更版本。 IntraServletContainerRenderer 在 servlet 转发中使用。EncodedImageRenderer 可用于渲染图层的动画图像。 CompositeRenderer 可用于请求只重新绘制带有变更数据的图层。这对于创建动画图层非常实用。有关详细信息，请参阅第 199 页第 11 章的*渲染涉及的考虑因素* 。

数据提供方

数据提供方是介于 MapJ 对象和地图数据之间的关键链接。每个作为 MapJ 组成部分的图层对象均有其自己的内部数据提供方。数据提供方用于访问数据源并返回向量数据。此外，在 MapJ 使用 LocalRenderer 渲染时也将调用数据提供方。

MapXtreme 中的数据提供方用于访问以下数据源：

- MapInfo 表
- 有空间选项的 Oracle
- Informix Universal Server SpatialWare DataBlade
- 有 SpatialWare 的 SQL Server
- JDBC 兼容的表，包含经度和纬度列
- ESRI Shapefiles
- 数据绑定 （将源自 TAB 文件的数据和 JDBC 数据源关联在一起）
- 栅格文件
- MapInfo 网格

MapJ 对象访问数据源的方式有如下两种：第一种方法是直接使用 LocalDataProviderRef 访问数据源。

第二种方法是请求 MapXtremeServlet 的实例来获取数据。 MapXtremeServlet 随后将使用 MapXtremeDataProviderRef 从其数据提供方服务器来直接访问数据源。由于 MapXtremeServlet 从数据源获取数据，它将数据以流方式传递回客户机的 MapJ 对象。MapXtremeServlet 采用的流数据传递方式不仅是极其高效的压缩方案。而且还可考虑数据所需的分辨率因素。例如，当数据用于渲染 640 x 480 图像时，数据可采用远远高于其存储分辨率的形式来传输。

每个与 MapJ 关联的图层均指定如何通过数据提供方引用来访问其底层数据源。LocalDataProviderRef 表示数据访问应该是 “本地”还是在包含 MapJ 的处理空间之内。MapXtremeDataProviderRef 说明 MapXtremeServlet 实例将在访问数据源时充当中介。

有关数据提供方的详细信息，请参阅第 167 页第 10 章的*在图层中绘制地图* 。

设计时的考虑因素

在规划地图绘制应用程序时切记以下因素。

客户端

- 是通过因特网部署，还是通过公司内网部署？
- 网络带宽是多少？
- 客户端是 applet 还是独立的应用程序？客户机是否需要类似 JDBC 驱动程序的附加软件或资源来运行？
- 是为特定的平台设计应用程序，还是为任意平台或混合环境设计应用程序？
- 用户将使用什么浏览器？是否需要 Swing 支持或其他浏览器插件？

- 客户端需要多少地图绘制功能？就具体需求而言，类似 AddThemeWizard 和 LegendContainer 的 JavaBean 有多大用途？

服务器端

- 正在构建的应用程序有多复杂？是否具备必要的硬件？
- 预期有多少用户使用相应应用程序？预期的用户峰值负载是多少？
- 需要从 Web 服务器和 / 或应用服务器获取什么服务？
- 对于正在构建的应用程序类型而言，是否具备适当的必要技巧？其中可能包括 Java 编程、数据库管理、 Web 开发等技术。
- 是否考虑到任何安全或网络问题？
- 应用程序需要与哪些其他软件交互？
- 要使用的 Java 是什么版本？所有组件是否支持共同的版本？

地图绘制概念

4

本章概要介绍使用 MapXtreme 创建地图绘制应用程序所要涉及到的众多概念。

本章内容：

组织数据和地图：表的概览

使用 MapXtreme Java 需要涉及到包含记录和地图的文件。数据既可采用 MapInfo 格式，也可采用支持空间数据的格式。本章将介绍 MapInfo 文件格式 (.tab)、地图定义，以及可以存储源自 JDBC 数据库的数据或 MapInfo 数据的基于 XML 的文本文件。

文件如何构成表

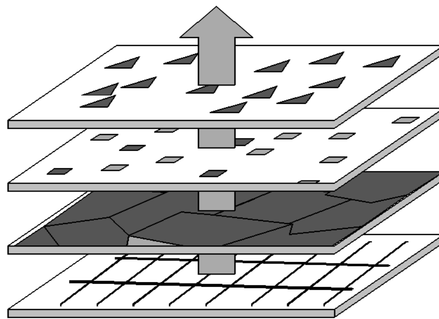
MapInfo .tab 文件可以组织成为一组用于构建图层的文件。 MapInfo 表包含以下类型的关联文件：

- *.tab：此文件说明 MapInfo 表的结构。这是一个说明数据文件格式的小文本文件。
- *.dat(.mdb, .dbf, .txt, .xls or .wks)：这些文件包含制表数据。
- *.map：此文件说明图形对象 （如果表没有地图对象，则该文件将不存在）。
- *.id：此文件是一个交叉索引文件，链接数据和对象 （如果表没有地图对象，则该文件将不存在）。
- *.ind: 这是一个索引文件。使用该索引文件，可搜索地图对象 （如果表没有索引，该文件将不存在）。

这些文件在一起构成 MapXtreme Java 中的单一 .tab 图层。

MapInfo 表和 MapXtreme 图层

每个可绘图 MapInfo 表均可在 MapXtreme 应用程序显示为地图中的图层。例如，籍此可以显示客户图层、街道图层和县界图层。



这些图层可视为透明，其中每个图层包含地图的不同组成部分。图层相互叠加在一起，以便可以看到地图的全貌。图层自下而上绘制。这些图层均在 “图层控制”对话框中列出，最上面的图层列在首位。

地图定义

地图定义说明了构成地图的数据，包括所要显示的数据、存储位置、数据组织为图层的方式、数据的显示方式 （如所用颜色、标注图元、地图缩放设置）。

MapXtreme Java 附带了多种涵盖全球地理和人口统计信息的示例数据。这些数据以两种格式提供：.mdf 和 .gst。（有关数据集的详细说明，请参阅产品 CD 上的 PDF 文档 MapXtreme Java Sample Data Set。）各种类型的说明如下。

MapInfo Geosets

MapInfo .tab 文件集称为 geoset。如果熟悉 MapInfo Professional，可以注意到 geoset 和工作空间的概念类似。工作空间是一个已保存的 MapInfo 表 (.tab) 和窗口的配置。无需打开单个的 .tab 文件，即可使用其特定的显示设置来打开 geoset (*somefile*.gst) 和所有图层。

但是， Geosets 在 MapXtreme Java 中的功能受到了限制。这些限制主要是指不可以保存到远程数据库的 MapInfo .tab 文件。

MapXtreme 不能打开 MapInfo 工作空间 （.wor 文件类型）。如果正在使用的是 MapInfo Professional，则可以使用 MapInfo Geoset 实用程序将工作空间另存为 geoset，然后可以使用 MapXtreme Java 管理器将其加载到 MapXtreme Java。

地图定义

MapXtreme Java 提供了比 geoset 更加出色的数据格式。地图定义是基于 XML 的文本文件，其中包括可以另存为文件或 JDBC 数据库记录的图层信息。这些文件可以向前兼容众多格式，并可轻松编辑，符合正在发展中的用于数据传输的 XML 标准。我们强烈建议使用地图定义而非 geoset 。

地图定义是使用 MapXtreme Java 管理器创建并随 MapXtreme Java 提供的文件。在*第 5 章：管理 MapXtreme Java* 中提供了有关的详细说明。在保存时，这些文件可以使用扩展名 .mdf 保存，存储为 JDBC 数据库记录，或是可以轻松检索的命名地图。

在 MapXtreme Java 3.x 中创建的地图定义将自动导入到 MapXtreme Java 4.x 之中。不支持在 3.x 之前的版本中创建的地图定义。此时需要创建新的地图定义，或者通过使用 MapXtreme Java 管理器，或者通过 MapDefContainer 接口以编程方式创建。有关详细信息，请参阅第 160 页。

注： MapXtreme Java 3.x 不能打开 4.x 版的地图定义。

要熟悉地图和 MapXtreme，可以在 MapXtreme Java 管理器中打开示例 .mdf。采用不同的图层设置进行实验 （有关详细信息，请参阅*第 5 章：管理 MapXtreme Java* ）。在准备就绪可以保存工作时，可将其另存为地图定义文件，在此既可另存为远程数据库表中的一条记录，也可另存为命名地图。

图元

MapXtreme 中的地图由地图对象的图层构成。这些地图对象可通过图元对象在 MapXtreme 中访问。此处有三种基本图元类型：

- 区域：特指覆盖给定区域的闭合对象。其中包括多边形、椭圆和矩形。区域包括国界、邮政编码边界和销售区域等。
- 点对象：表示数据的单一位置。其示例包括客户位置、饭店和停车计时器等。
- 线对象：覆盖给定距离的开放对象。包括直线、折线和弧线等。其示例有街道、河流和电力线路等。

各种不同类型的对象置于单独的图层之中 （最常见），或者也可以在同一图层中组合不同的对象。借助于 MapXtreme，可以自定义和显示这些对象，令地图满足具体的需求。

样式

MapXtreme Java 地图中的所有图元均显示特定的可见特征，例如颜色或符号类型。相应的特征也称为样式。 MapXtreme Java 包括使用 Java2D API 渲染能力的众多显示属性，其中有直线和区域的符号刷、虚线和平行线、矢量符号和缩放符号和标记。

样式可以通过编程设置，也可以通过 MapXtreme Java 管理器的图层控制对话框设置。有关样式 API 的详细信息，请参阅*第 14 章：标注和样式* 。有关图层控制的详细信息，请参阅*第 5 章：管理 MapXtreme Java* 。

标注

标注是 MapXtreme Java 中功能强大的特性之一，可用于增强地图功能，并向用户提供正确的消息。标注不仅仅局限于添加文本来说明地图的特性。使用 MapXtreme Java，可以控制字体类型、大小、颜色、位置，并使用光晕和轮廓线等富有创意的效果，为每个图层创建不同的标注。标注内容可以通过从图层数据源的一个或多个列取数据的表达式来生成。如果需要区分相同图层中的标注，还可以创建标注专题。

有关标注的详细信息，请参阅*第 14 章：标注和样式*。

地图数据分析

分析地图以进一步了解行列格式数据之外的更多信息，也是 MapInfo 公司绘图软件的强大功能之一。通过在地图上显示数据，可以对数据进行可视化的比较，这将有助于作出更出色的业务决策。

MapXtreme Java 提供了分析地图的众多方式，从使用地图工具到点击图元，创建显示图元相互关系的专题地图，乃至根据自定义条件搜索底层数据不一而足。

将数据制为地图

借助于数据提供方，可从其他来源向地图添加数据。例如，如果有一个按照县划分销售额的 Oracle 空间数据库表，则可以将相应数据显示在地图上，按照县查看销售模式的点趋势图。

MapXtreme 支持访问多种不同类型的数据源，其中包括：

- Oracle Spatial
- Informix Universal Server SpatialWare DataBlade
- SQL Server with SpatialWare
- JDBC 兼容的数据库 （存储在 X,Y 列的空间数据）

此外， MapXtreme Java 支持数据绑定，即实现了将 .tab 文件中的数据关联到 JDBC 数据以构成图层。

有关从这些数据源添加数据的详细信息，请参阅*第 10 章：在图层中绘制地图*。

5

管理 MapXtreme Java

本章介绍使用 MapXtreme Java 管理器如何管理数据和命名资源。

本章内容：

MapXtreme Java 管理器

MapXtreme Java 管理器是一个 Java 服务器和客户机 GUI，可用于全方位管理各种 MapInfo 企业级产品，其中包括 MapXtreme Java 地图服务、 MapMarker J Server 地图编码服务和 Routing J Server 线路规划服务。目前上述服务中只启用了 MapXtreme Java 服务，但是用户可以自行添加地理编码和线路规划服务。有关详细信息，请参阅第 65 页的*向 MapXtreme Java 管理器添加服务* 。

MapXtreme Java 管理器服务器和客户机在相互之间通过 XML 通信。客户机可以作为应用程序或浏览器中的 applet 运行。作为应用程序运行时可采用 Web 服务器模式或单机模式。MapXtreme Java 管理器客户机可用于：

- 管理和配置地图图层 （地图定义面板）
- 管理命名资源，其中包括地图、图层和样式 （命名资源面板）
- 快速构建原型 Web 应用程序 （Web 应用程序面板）。
- 管理 JDBC 连接 （连接管理器面板）。

关于 MJM 元素

MapXtreme Java 管理器 (MJM) 使用预置组件执行特定的地图绘制任务。地图定义面板上的按钮均为 JavaBean，可添加至用户自行开发的应用程序。有关详细信息，请参阅*第 7 章：MapXtreme JavaBeans* 。

Web 应用程序构建器使用 Java 服务器页技术实现快速的原型开发。定制标记库中的 JSP 标记也可供用户自行使用。请参阅*第 6 章: Web 应用程序构建器*和*附录 A: 定制 JSP 标记库* 。

运行 MapXtreme Java 管理器

由于 MapXtreme Java 管理器需要与 Web 应用服务器通信，因此在启动之前，确保 Web 应用服务器 （如 Apache/Tomcat）正在运行。采用单机模式运行 Manager 无需启动应用服务器，但是将无法使用某些功能 （如 Web 应用程序构建器向导）

要运行 MapXtreme Java 管理器服务器，请执行以下操作：

- 启动 Tomcat。对于 Windows 用户，从 “程序”菜单或从 tomcat/bin 目录下运行 startup.bat 文件均可启动 Tomcat。 UNIX 用户可运行 startup.sh。

在 MapXtreme 服务器运行之后，可启动 MapXtreme Java 管理器客户机，将其作为 applet 或应用程序运行。

要启动 MapXtreme Java 管理器客户机作为 applet 运行，可执行以下操作：

- 打开 Web 浏览器，然后指向 MapXtreme Java 管理器 servlet 的 URL。例如：http://stockholm:8080/mapxtreme47/manager

要启动 MapXtreme Java 管理器作为应用程序运行，请执行以下操作：

- 在 Windows 环境下，从“程序”菜单选择 MapXtreme Java 管理器快捷方式。对于其他环境，或者要从命令行来进行启动，可使用以下语法。

```
java com.mapinfo.mjm.client.MJMClient <MJM Servlet URL>
```

安全考虑因素

如果部署的是直接和 MapXtremeServlet 对话的胖 MapXtreme Java 客户机应用程序，则由于处于同一上下文中，该应用程序还可以访问 MapXtreme Java 管理器 servlet。如果在防火墙之外公开 mapxtreme47 servlet 的上下文，那么这将会同时公开 MapXtreme Java Manger。某些访问者通过访问 http://host/mapxtreme47/manager，即可浏览文件系统甚至是保存在该处的文件。

对于这些胖客户机，只需配置 Web 服务器，不允许访问 /manager servlet 即可解决上述问题。

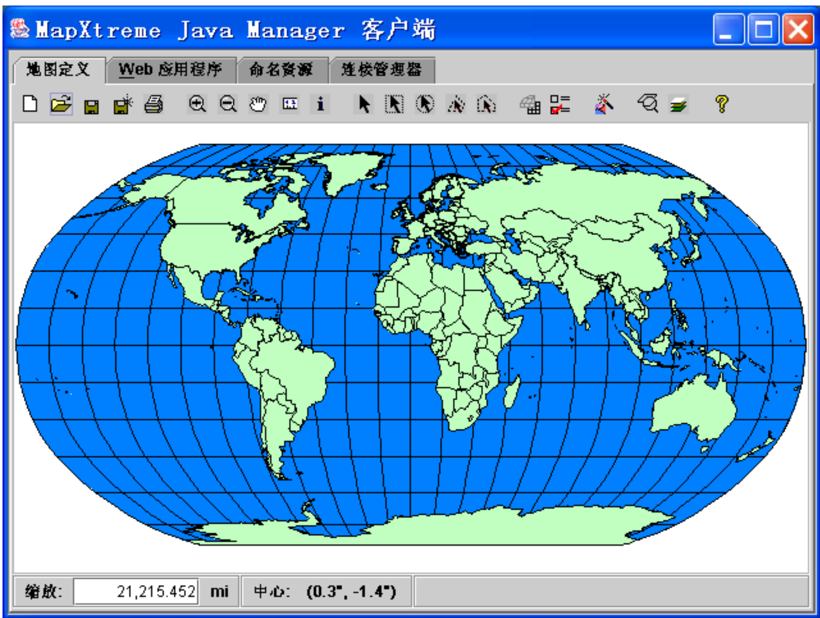
地图定义面板

在地图定义面板中可构建地图。从地图定义面板还可以：

- 以两种格式加载和保存地图图层：mdf （地图定义）或 .gst (Geosets) 格式
- 从文件、数据库或命名资源添加图层，并编辑其属性设置
- 平移和缩放，更改显示和标注设置
- 对地图上显示的图层重新排序
- 创建专题图层，并保存所有设置，用于在必要时加载到应用程序

默认情况下，所有图层均由 MapXtreme Java 管理器使用 MapXtremeDataProviders 创建；但是也可选择使用 MapXtremeDataProviders 或 LocalDataProviders 来创建。

下图显示了已加载地图定义的 MapXtreme Java 管理器客户机界面。要加载或创建地图定义，请参阅第 68 页的*加载现有的地图定义*中的说明。



Web 应用程序面板

MapXtreme Java 管理器的 Web 应用程序面板为开发人员提供了向导，指导开发人员轻松快捷地构建定制 Web 应用程序。借助于 Sun 的 Java 服务器页技术这一 Java 2 平台企业版的关键组件，开发人员使用这一快捷原型即可独立于平台，创建和维护 Web 页面中的静态和动态内容。 JSP 技术将用户界面和内容生成隔离开来，设计人员借助于此，更改整体页面布局时即无需更改底层的动态内容。

这一原型构建器使用的标记类似于 JSP XML 定制标记，其行为类似于窗口部件，只需从列表中选择即可将其添加到构建器的布局框架中。我们在 JSP 库 （将在*附录 A: 定制 JSP 标记库*中详细讨论）中提供了各种典型绘图插件，例如工具栏、地图、图层控件和图例插件。此外还包括有关说明，说明如何构建定制 JSP 标记，并将其添加到向导以增大从中构建应用程序的插件池。

有关使用 Web 应用程序面板的说明，请参阅*第 6 章：Web 应用程序构建器* 。

命名资源面板

MapXtreme Java 管理器的命名资源面板可用于管理已经给定唯一名称或别名的资源。在此可创建用于命名地图和命名图层的上下文，并从安装期间创建的 mistyle 上下文访问命名样式。有关命名资源的详细信息，请参阅第 75 页的*保存地图定义*。

连接管理器面板

连接管理器提供了管理 JDBC 连接的用户界面 （如编辑 miconnections.properties 的内容）。这些连接由 miconnections.properties 文件处理，并在连接管理器面板上管理。有关详细信息，请参阅*第 12 章：访问远程数据*。

向 MapXtreme Java 管理器添加服务

采用标准安装，将自动为 MapXtreme Java 管理器预配置有关 MapXtremeServlet 服务的信息，并且该服务可以直接使用 （如使用 Web 应用程序向导创建的应用程序将使用 MapXtremeServlet 来绘制地图）。

MapXtreme Java 管理器还可以配置有关于其他服务的信息，例如地理编码 （通过 MapMarker J 服务器）或线路规划 （通过 Routing J 服务器）。但在利用相应的其他服务之前，必须手动配置 MapXtreme Java 管理器，如下所示。

注： MapMarker J 服务器和 Routing J 服务器均是可从 MapInfo 公司购买的产品。

向 MapXtreme Java 管理器添加 MapMarker 支持

在配置的 MapMarker 服务器有效工作之后，可通过将以下初始化参数添加到 webapps/mapxtreme47/WEB-INF/web.xml 的 MapXtreme Java 管理器 servlet 部分，令 MapXtreme Java 管理器意识到该服务器的存在。

```
<init-param>
          <param-name>mapmarker</param-name>
          <param-value>mapmarker://testbed:4141</param-value>
</init-param>
```

该参数的名为 mapmarker。注意其参数值是一个伪 URL （起始为 mapmarker:// 而非 http://）。使用这一 URL 指定 MapMarker 服务器的主机和端口 （中间用冒号分隔）。

预计下一版本的 MapMarker 可以使用常规 HTTP URL，而不需要再使用主机和端口号。在将来的 MapMarker 版本发布之后，即可使用 MapMarker 服务器的 http:// 常规 URL 来取代 mapmarker:// 的伪 URL。

重新启动 servlet 容器。再次运行 MapXtreme Java 管理器时， Web 应用程序向导将包括附加的插件 （如添加到工具栏插件中的按钮），这些插件可用于向 JSP 应用程序添加地理编码支持。

向 MapXtreme Java 管理器添加 Routing J 服务器支持

在配置的 Routing J 服务器有效工作之后，可将以下初始化参数添加到 webapps/mapxtreme47/WEB-INF/web.xml 的 MapXtreme Java 管理器 servlet 部分，令 MapXtreme Java 管理器意识到路线规划服务器的存在。

```
<init-param>
       <param-name>routing</param-name>
       <param-value>http://testbed:85/routing/servlet/
              RoutingServlet</param-value>
</init-param>
```

该参数的名为 routing。其参数值是指向 Routing J 服务器 servlet 的 URL。

重新启动 servlet 容器。再次运行 MapXtreme Java 管理器时， Web 应用程序向导将包括附加的插件 （如添加到工具栏插件中的按钮），这些插件可用于向 JSP 应用程序添加路线规划操作。

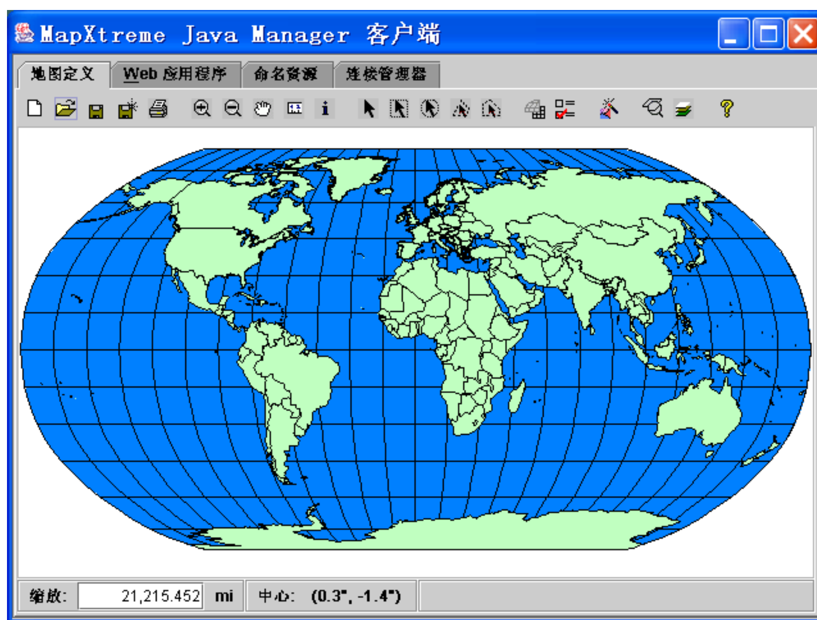
使用 MapXtreme Java 管理器管理数据

管理要在地图中显示的数据是 MapXtreme Java 管理器的主要功能之一。 MapXtreme Java 将此数据显示为图层集合，每个图层 （理想情况）均由一种类型的图元构成，例如行政边界、客户位置、无线网络或高速公路等图元。

这些图层可以保存，并可通过使用 MapXtreme Java 管理器创建地图定义得以复用。地图定义说明将要纳入地图的地图图元以及相应的显示设置。地图定义是基于 XML 文本，提供有关图层集合的说明。地图定义可作为远程数据库中的记录或命名资源档案库中的命名地图，保存为 .mdf 格式。

使用 MapXtreme Java 管理器来创建适当的基础地图，以便用作地图的参考信息。例如，可能需要区域边界、街道图层和商店位置图层以资参考。

此外，还需要包括和应用程序相关的特定地图数据，例如客户位置、基站位置或送货路线。任何记录只要具有类似客户地址的位置组件，即可用作地图数据，。



构建地图集，可从加载 MapXtreme Java 附带的预定义地图定义 （.mdf 格式）作为开始。有关详细说明，请参阅第 68 页的*加载现有的地图定义* 。 /examples/server/data/machine-name 目录中还提供了一个示例 world.mdf。

注： 示例基础地图数据集的说明位于联机 PDF 文档 MapXtreme Java Sample Data Set 之中，该文件可见于产品 CD 或安装产品之后的 /maps 目录之下。

在打开某些图层之后，可自定义显示图层的方式，并可增减图层或对其重新排序。有关图层自定义工具的说明，请参阅第 77 页的*使用地图工具控制图层* 。

在对图层显示特征满意之后，可将其保存为地图定义，用于在必要时加载到应用程序之中。

Geosets 和地图定义

本节提供有关 geosets 和地图定义的说明。

Geosets

MapXtreme Java 的示例数据集包括 MapInfo 的 .tab 格式文件，这些文件划分为概念类似于工作空间的 geoset （扩展名为 .gst）组。例如，示例数据中有一个 world.tab 文件和一个 world.gst 文件。 geoset World.gst 是一种元数据文件，用于说明一个包括 world.tab 和其他文件的 .tab 文件集合。

Geosets 是一类可以加载到 MapXtreme Java 管理器中的文件。但是需要注意的是，这些文件限定为 MapInfo .tab 格式，不能存储到远程数据库或命名资源档案库中。此外， geosets 中的样式不能更改。

地图定义

为了打破 geosets 的限制条件， MapXtreme Java 还提供了采用 .mdf 格式的示例数据集。基于 XML 的地图定义是说明地图图元和设置的文本文件。地图定义可以保存为文件 （扩展名为 .mdf）或存储为 RDBMS 中的记录，或存储为命名资源档案库中的命名地图。我们强烈建议使用地图定义。如果使用 geosets，那么可在使用之后将地图设置另存为地图定义。

可以在 MapXtreme Java 3.x 中打开的地图定义将会自动导入到 MapXtreme Java 4.x 之中。将不再支持 3.0.x 之前的版本中创建的地图定义。要创建新的地图定义，必须通过使用 MapXtreme Java 管理器或者通过 MapDefContainer 接口以编程方式来创建 （请参阅第 160 页的*通过编程保存地图*）。

加载现有的地图定义

用户可以打开采用以下 4 种方式存储的地图数据：

- 存储为 .mdf 或 .gst 文件
- 存储为命名地图 （此前给定唯一名称的图层集合和设置）
- 存储为最近访问的地图定义 (MRU)
- 存储在远程数据库中

后续说明对上述各种存储方式分别作出了解释。

要显示现有地图定义，可执行以下操作：

1. 按照第 62 页的*运行 MapXtreme Java 管理器*中的说明，运行 MapXtreme Java 管理器 Servlet 和客户机。此时将显示地图定义面板。
2. 单击地图定义面板上的打开按钮。此时将显示加载地图定义屏幕。其中提供了以下选项卡：文件、最近使用的、命名和数据库。
3. 要加载一个存储为文件的地图定义，可单击文件选项卡，并导航至存放 .mdf 或 .gst 文件的位置。单击加载。

   注： 相应文件存储在 MapXtreme Java 服务器上，因此即使正在将 MapXtreme Java 管理器作为 applet 在远程计算机上运行，也可以导航至服务器的相应目录。无需在该服务器计算机上即可访问地图文件。

4. 要加载此前打开过并且保存到 "最近使用的" 选项卡中的地图定义，可单击最近使用的，然后选择适当的地图定义，再单击加载。
5. 要加载此前的命名地图，可单击命名选项卡。如有必要，可在左侧窗格中单击适当上下文，导航至命名地图的存储位置。突出显示资源列表中的地图，然后单击加载。（有关创建命名地图的说明，请参阅第 98 页的*命名地图*。）
6. 要从 RDBMS 加载地图定义，可单击数据库选项卡。



在连接框中，从此前保存的连接 （如有）列表选择连接。在将地图定义保存到命名连接之前，必须先定义该命名连接。有关创建命名连接的详细信息，请参阅*第 12 章：访问远程数据*。

7. 在表或查询组中，从以下选项中选择：

   使用 **MAPINFO.MAPDEFINITIONS** 表 （默认）

   使用表：提供表名、名称列和地图定义列。

   使用 **SQL**：自行提供 SQL 查询语句

8. 在地图定义组中，单击刷新按钮以显示现有地图定义的列表。从中选择一个地图定义文件，然后单击加载。此时将显示一个地图定义，用户可直接对齐进行自定义操作。

创建地图定义

下一节说明如何使用 MapXtreme Java 管理器创建地图定义。

使用增加图层向导

地图定义可以通过添加新图层，或采用新设置和新名称保存现有地图定义或 geosets 来创建。 MapXtreme Java 提供了增加图层向导来帮助用户完成相关的操作。每次可在一个图层上构建地图定义。

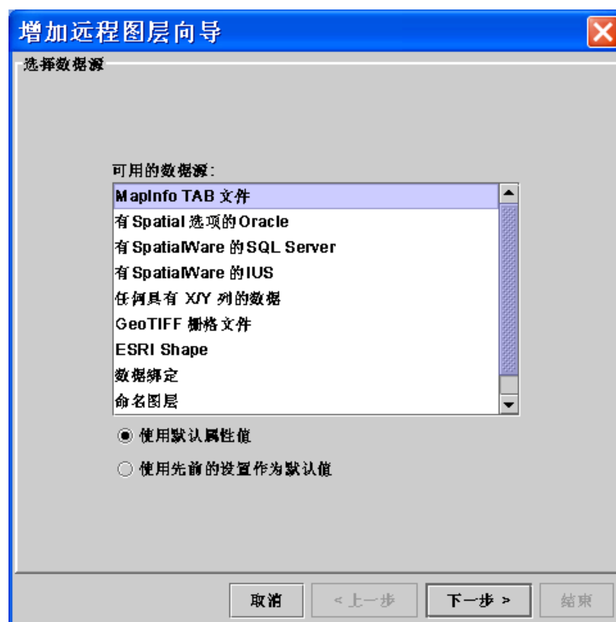
注： 在增加图层向导中，“远程”表示可能潜在地浏览远程计算机，尽管数据可能就在本地。

以下步骤说明使用该向导如何从基于文件的数据源、 JDBC 数据源、命名图层和数据界限图层添加新图层。要从现有的地图定义或 geoset 创建新的地图定义，可使用 “另存为 ...” 按钮。

添加 MapInfo TAB 文件和其他基于文件的地图数据

要使用基于文件的数据创建新的地图定义，可执行以下操作：

1. 从地图定义面板，单击新建按钮。从地图中删除任意现存的图层都将留下一个空地图。单击图层控制按钮。
2. 从 “图层控制”对话框，单击添加。此时将显示增加远程图层向导。
3. 要添加 MapInfo .tab 文件，可从 “选择数据源”面板提供的列表中选择 **MAPINFO TAB** 文件。单击下一步。



4. 在 “指定 MapInfo 表信息” 对话框中，单击添加按钮，导航至文件位置，然后选择一个或多个文件。单击打开以返回到增加图层向导，然后单击下一步以继续。
5. 在 “指定其他图层设置” 对话框中，根据需要可提供图层名称。
6. 单击完成。此时将返回到 “图层控制” 对话框。该对话框的图层 / 专题列表中显示了 .tab 文件。单击确定以查看地图中的图层。
7. 通过设置显示、图层属性和标注属性、重排图层显示方式、增减图层以及图层是否可见或包含标注，自定义所需图层。有关详细信息，请参阅第 83 页的*图层控制* 。
8. 在对设置感到满意之后，单击确定以离开 “图层控制” 对话框。此时将显示新建的地图。如有必要，返回到图层控制面板，更改显示设置或返回到增加图层向导以添加更多图层。

   在完成创建地图之后，需要将其另存为地图定义。请参阅第 75 页的*保存地图定义* 。

注： 上述操作步骤适用于类似的添加 **GeoTIFF Raster** 或 **ESRI Shape**。要添加 **MapInfo** 网格文件，可选择 TAB 数据源，然后打开与所要添加的网格文件关联的 .tab 文件。

添加 JDBC 地图数据

要使用 JDBC 数据创建新的地图定义，可执行以下操作：

1. 从地图定义面板，单击新建按钮。从地图中删除任意现存的图层都将留下一个空地图。单击图层控制按钮。
2. 从“图层控制”对话框，单击添加。此时将显示增加图层向导。
3. 从列表选择 JDBC 数据源，例如有空间选项的 ORACLE。选择默认属性值或此前的设置。单击下一步。
4. 输入用于数据源的连接信息。
5. 在用于数据源的“指定表或查询”面板中，选择添加表或可选查询数据库。单击下一步。
6. 指定 MapXtreme Java 是否应该查询 MAPINFO_MAPCATALOG 以获取相关设置或自行执行设置。单击下一步。

   如果图元或标注级别的样式信息存储在表中，可从其存储的 MAPCATALOG 中指定信息的存储方式和样式列的名称。默认行为为无。单击下一步。
7. 在指定其他图层设置面板，指定图层名（可选）。单击完成。此时将返回到“图层控制”对话框。您所添加的图层将显示在列表中。
8. 重复执行上述步骤可添加更多 JDBC 图层。
9. 通过设置显示、标注属性、重排图层显示方式、增减图层以及图层是否可见或包含标注，自定义所需图层。有关详细信息，请参阅第 83 页的*图层控制*。
10. 在对设置感到满意之后，单击确定以离开“图层控制”对话框。此时将显示新建的地图。如有必要，返回到“图层控制”对话框，更改显示设置或返回到增加图层向导以添加更多图层。

    在完成创建地图之后，需要将其另存为地图定义。请参阅第 75 页的*保存地图定义*。

添加命名图层

命名图层是此前使用唯一名称保存的图层。您可以通过增加图层向导，象检索其他任何图层一样检索命名图层。有关创建命名地图的说明，请参阅第 99 页的*命名图层*。

要添加命名图层，请执行以下操作：

1. 从地图定义面板，单击新建。从地图中删除任意现存的图层都将留下一个空地图。单击图层控制按钮。
2. 从“图层控制”对话框，单击添加。此时将显示增加图层向导。
3. 从“选择数据源”面板提供的数据源列表中选择命名图层。单击下一步。
4. 单击选择命名图层按钮，显示命名资源对话框。在左侧窗格中，单击包含命名图层的上下文。在资源面板中突出显示命名图层，然后单击加载。

5. 在 “选择命名图层”面板中，单击完成。此时将返回到 “图层控制”对话框，该框中显示了所添加的图层。单击确定以查看地图中的图层。
6. 继续添加其他图层来构建地图 。通过设置显示、标注属性、重排图层显示方式、增减图
   层以及图层是否可见或包含标注，自定义所需图层。有关详细信息，请参阅
   第 83 页的*图层控制* 。
7. 在对设置感到满意之后，单击确定以离开 “图层控制”对话框。此时将显示新建的地图。如有必要，返回到 “图层控制”对话框，更改显示设置或返回到增加图层向导以添加更多图层。

   在完成创建地图之后，需要将其另存为地图定义。请参阅第 75 页的*保存地图定义* 。

添加数据绑定图层

数据绑定图元允许您通过联接 MapInfo .tab 文件中的图元几何对象与 JDBC 数据库中的属性信息来构建地图。

要通过添加数据界限图层来创建新的地图定义，可执行以下操作：

1. 从地图定义面板，单击新建按钮。
2. 从 “图层控制”对话框，单击添加。此时将显示增加图层向导。
3. 从可用数据源列表中选择数据绑定。单击下一步。
4. 在指定 MapInfo 表信息面板上，单击 **...** 按钮，导航至 .tab 文件位置。单击打开以返回到增加图层向导，然后单击下一步以继续。
5. 指定绑定图层数据源信息 （JDBC 驱动程序类名、连接 URL、用户名和口令）。单击下一步。
6. 提供您要使用 MapInfo .tab 几何信息构建的表或查询。可选择提供用于该表的所有者名称或用于查询的 ID 列。单击下一步。
7. 在 “地理和连接图层列”面板，输入用于地理图层 （MapInfo .tab 文件）的列名，然后单击添加。对于源自 JDBC 数据源的联接图层，可指定和地理列相关的列名，然后单击添加。如果要使用附加列进行联接，可在添加相应的列。单击下一步。
8. 可以选择给出图层名称。单击完成。此时将返回到 “图层控制”对话框。数据绑定图层将显示在图层 / 专题列表中。单击确定以查看地图中的图层。
9. 通过设置显示、标注属性、重排图层显示方式、增减图层以及图层是否可见或包含标注，自定义所需图层。有关详细信息，请参阅第 83 页的*图层控制* 。
10. 在对设置感到满意之后，单击确定以离开 “图层控制”对话框。此时将显示新建的地图。如有必要，返回到 “图层控制”对话框，更改显示设置或返回到增加图层向导以添加更多图层。

    在完成创建地图之后，需要将其另存为地图定义。

创建分析图层

您可以使用增加图层向导创建饼图或条形图作为分析图层。要创建分析图层，请执行以下操作：

1. 从地图定义面板，单击打开按钮。打开要向其添加分析图层的文件。单击图层控制按钮。
2. 从 “图层控制”对话框，单击添加。此时将显示增加图层向导。
3. 从可用数据源列表中选择分析图层。单击下一步。
4. 在 “指定图层和要分析的列”面板，选择图层、要分析的列和分析类型。并行条形图垂直显示，而堆叠条形饼图将会水平显示。单击下一步。
5. 在 “指定饼图属性”面板，选择插件颜色、字符大小、方向和位置。单击下一步。



如果已在第 4 步中选择了一个条形图类型，则选择条块颜色、字符大小和方向。单击下一步。



6. 在 “指定其他图层设置” 对话框，指定新图层的名称。单击 “完成”。

保存地图定义

保存地图定义时，有若干种不同的选择。

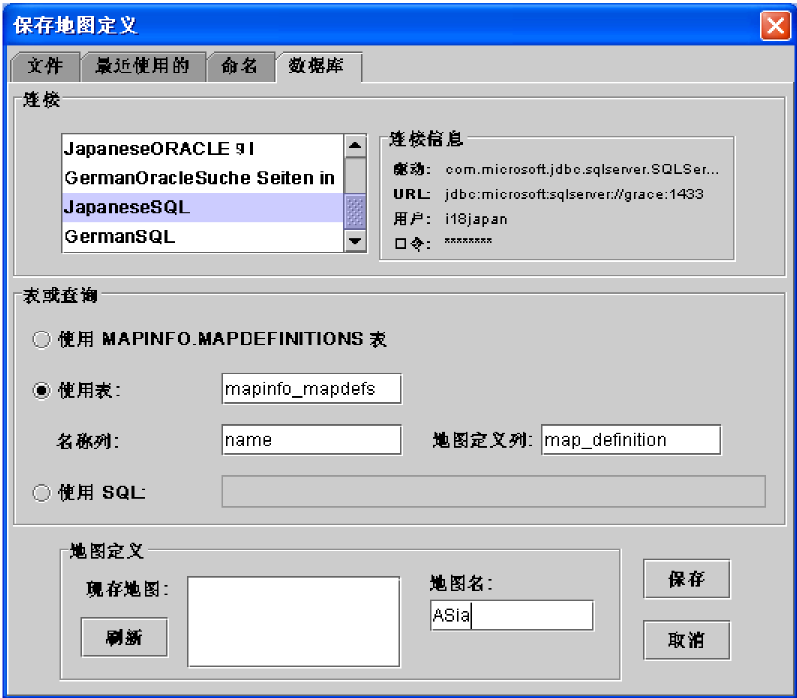
此时既可保存到文件，保存为命名地图，也可保存到远程数据库。

1. 要保存地图，可从地图定义面板单击保存按钮。此时将显示带有 4 个选项卡的地图定义对话框。文件、最近使用的、命名和数据库。转至适合于您要保存的地图定义类型所在的适当选项卡。
2. 如果要保存到文件，可单击文件选项卡，提供文件名和存储文件的位置。单击保存。地图将存储为 .mdf 格式。
3. 如果要将地图保存为命名地图，可从 “命名” 选项卡，单击要将地图定义保存到的上下文。为 “命名” 框中的地图定义提供名称。单击保存。相应信息现在将保存为指定上下文中的 XML 格式的地图定义。

   如果没有保存命名地图的上下文，则必须设置相应的上下文。转至 MapXtreme Java 管理器客户机的命名资源面板。有关详细信息，请参阅第 97 页的*管理命名资源*。



4. 要将地图定义保存到远程数据库，可单击数据库选项卡。



5. 在 “连接”框中，从此前保存的连接 （如有）列表选择连接。在将地图定义保存到命名连接之前，必须先定义该命名连接。有关创建命名连接的详细信息，请参阅 *第 12 章： 访问远程数据* 。

6. 在表或查询群组中，从以下选项中选择：
   使用 **MAPINFO.MAPDEFINITIONS** 表（默认）
   使用表：提供表名、名称列和地图定义列
   使用 **SQL**：提供自己的插入 / 更新 SQL 语句
7. 在地图定义群组中，指定地图名，然后单击保存。相应信息现在将保存为数据库中 XML 格式的地图定义。

使用地图工具控制图层

在 MapXtreme Java 管理器中打开地图图层之后，可以根据具体需要自定义其显示。使用地图工具栏上的按钮来控制地图的显示。

我们将在此介绍以下组件：放大、缩小、平移、标尺、信息、选择工具、首选项、地图选项、创建专题和图层控制。

放大工具



使用放大工具可获取地图或图层的较近的区域视图。

要使用放大工具，可在要放大的区域中心处点击 “放大 “光标。这一操作将通过两个线性因素放大该区域。该点将位于放大视图中的地图中心。重复此操作步骤，直至得到适当的放大级别。

要放大一个矩形区域，可在地图或图层上，通过对角拖动放大光标画出一个选取框。选取框中的区域即可放大。

缩小工具



使用缩小工具可获取地图或图层的较宽的区域视图。

要使用缩小工具，可在要缩小的区域中心处点击缩小光标。这一操作将通过两个线性因素放大该区域。该点将位于缩小视图中的地图中心。

平移工具



使用平移工具将地图重新定位于其窗口之中。

要移动或调整地图显示，可单击地图中的区域，同时按住鼠标按钮，以适当的方向拖动地图。在释放鼠标按钮之后，地图将在其新的位置上重画。

标尺工具



使用标尺工具可测量两点之间的距离和多点之间的总距离。

要使用标尺工具，可单击地图上的起点。双击终点。标尺窗口将显示两点之间的距离。

要显示一条路线的中间点的距离，可单击起点，然后继续单击其他点。“标尺”窗口将显示最后一段距离和总计的距离。要结束该操作，可双击所要测量的最后一点，或按下 Esc 键。

信息工具



信息工具提供了与存在于给定点的地图对象关联的属性视图。

要使用信息工具，可单击地图图元。此时将显示信息工具窗口。

如果选中的是用于唯一图层的对象，则 “信息工具”窗口将显示与该点处的地图对象关联的属性。用于该记录的每个属性均可查看。要查看完全列表，需要放大该窗口。

如果地图包含多个图层，“信息工具”窗口将显示图层列表。突出显示图层，然后双击即可查看图层级别的属性信息。单击 “列表”按钮以返回到图层列表。

选择工具

通过 MapXtreme Java，可选择地图图层中的一个或多个图元，以在其上执行附加的操作。MapXtreme Java 管理器提供 5 种选择工具。选择工具可用于选择单独的图层。其他 4 个工具在所选图元区域中提供不同的限定区域 （矩形、半径、多边形、范围）。

5 个选择工具公共的行为是可通过使用 Shift 键添加所选图元的操作。例如，单击选择工具以选择图元，按住 Shift 键同时单击另一个要包含在第一个选择中的图元。

所有图元均从地图最顶部的可选图层中选择。在默认情况下，图层是不可选择的。为此必须在对应于所要选择的图层的 “图层控制”对话框中选中 “可选”框。

在选中图元之后，可将其保存为选择专题。有关选择专题详细信息，请参阅*第 15 章：专题地图绘制和分析* 。

选择工具



要使用选择工具，可通过首选方法选取 “选择工具”，然后单击要选的图元。按住 Shift 同时单击以选择附加图元。

矩形选择工具



要使用 “矩形选择”工具，可选择工具，然后单击并拖动鼠标，在要选择的对象周围形成一个矩形。按住 Shift 键同时单击鼠标，在此拖动鼠标以选择要包含在第一个选择中的、位于另一个矩形中的图元。

半径选择



要使用 “半径选择”工具，可选择工具，然后单击并拖动鼠标，在要选择的对象周围形成一个圆。按住 Shift 键的同时单击鼠标，创建另外一个圆，将所选图元添加到第一个选择中。

多边形选择工具



要使用 “多边形选择”工具，可选择工具，然后单击地图以开始创建多边形，即在要选择的图元周围形成一个多边形。继续单击附加的点以形成多边形。通过在起点附近单击或双击来完成多边形。务必确保该多边形具有至少三个节点。

范围选择工具



钥匙 “范围选择”工具，可选择工具，单击边界对象，选择其中所有图元。按住 Shift 键的同时单击其他范围，将所选图元添加到该几何对象。

其他地图工具

本节说明可用的其他工具，例如首选项和地图选项。

首选项



单击 “首选项”按钮设置默认的地图目录、 MIME 类型 （默认为 image/gif），以及 MapXtreme Java 是进行远程 （默认）还是本地渲染。

本地渲染主要用于测试目的，例如确认应用程序可有效完成本地渲染工作。如果客户机不具备可用于应用程序或 applet 的所有必须资源，则不应使用。例如，如果客户机没有安装必需的字体，则某些地图可能无法正确显示。



地图选项



要更改距离单位设置和投影，可以使用地图选项按钮。



从地图选项对话框的下拉菜单中选择距离单位。

要更改坐标系，可单击显示坐标系下拉菜单，显示选择坐标系对话框。

坐标系数据存储在一个名为 **micsys.txt** 的投影文件中。此文件列出了数百种支持的坐标系以及其相应的定义参数。

文件 micsys.txt 包含在 micsys.jar 之中。当 MapXtreme Java 管理器作为应用程序运行时，micsys.jar 将从 MapXtreme 的 lib/common 命令中加载。当 MapXtreme Java 管理器作为 applet 运行时， micsys.jar 将从 servlet 容器的 webapps/mapxtreme47/client 目录中加载。



查看整个图层按钮



使用 “查看整个图层”按钮可查看所选图层的范围。在 “查看整个图层”对话框中选择图层，然后单击 “确定”。地图在重新计算的缩放级别重画，以便所选图层的范围在地图上可见，供您查看整个图层。

注：　您可能无法使用 “查看整个图层”来查看查询图层的范围。如果图层定义为查询，并且该图层没有 QueryBuilder，则尝试在该查询图层上执行 “查看整个图层”时，将会看到错误消息。

添加图层工具



增加专题向导是一个向导工具，用于帮助您将图元专题或标注专题添加到地图。

对于图元专题，可以创建范围主题，根据一段范围内的值来划分图元；也可以创建单值专题，根据值来以影线表示图元。

该专题可以基于任何支持的列和当前地图中的图层。目前支持基于数字、字符串和日期列时间创建专题，以及基于点、线和区域图层创建专题。



部分操作将用于创建默认的专题图例，与新专题相关联。图例可以自定义，更改标题、字体、插入信息、说明文本和颜色。

对于标注专题， MapXtreme Java 支持以下三种类型：范围、单值和选择。例如，要区分同一图层中的主要和次要城市，可使用范围标注专题根据其相应的数据来对城市分组，例如人口数据。

有关标注专题的详细信息，请参阅*第 14 章：标注和样式* 。

有关添加专题向导的详细信息，请参阅第 116 页的 *AddTheme Bean* 。

图层控制

对于地图显示的真正控制能力存在于 “图层控制”对话框之中。使用其中的选项，可显示、删除、添加、编辑、选择、缩放图层和标注图层。您还可以更改地图图层和专题的顺序。

要访问 “图层控制”对话框，可单击地图工具栏上的图层控制按钮。



“图层控制”对话框显示了构成当前地图的所有图层和图层属性的状态。这些属性如下所示：可视性、可选和自动标注。每个复选框列上的图标都表示了相应属性。使用这些复选框，可以轻松更改一个或多个图层的属性。

在此还提供了更改显示、标准和专题设置的选项，以及重排、添加、编辑或移除图层的选项。保存地图定义，即可保留这些设置。此外还可以通过图层控制对话框的 “保存”按钮将图层保存为命名图层。

后续内容讨论 “图层控制”对话框中的特性。

图层／专题可视性

“图层控制”对话框中的可视性属性控制图层或专题在地图上是否可见。例如，要令图层不可见，可清除对图层的 “可视性”复选框所做的选择。相应图层将不再显示在地图中（但在保存地图定义时，其中仍然包括该图层）。如果要关注的只是多图层地图中的一个或多个图层，则这一特性将会尤为实用。

图层可选择性

通过 MapXtreme Java，可以选择用于控制和分析的地图对象。 “可选”复选框用于控制图层的地图对象是否可供选择。默认情况下，已添加到地图的图层为不可选。

这一特性在要从较低图层选择对象时非常实用。关闭要从中进行选择的图层上的所有图层的可选状态。

自动标注

如果用于标注的缩放级别设置适当，则选中自动标注复选框的任意图层都将显示标注。对于不希望显示标注的任意图层，可清除复选框中所做选择。默认情况下，添加到地图的图层不是自动标注的。

有关设置和控制标注显示的详细信息，请参阅第 92 页的*经由图层控制的标注选项*。

重排图层和专题

图层和专题按照其在 “图层控制”对话框中列示的顺序显示，地图最先绘制底层图层，最后绘制顶层图层。图层正确排序才可以避免意外隐藏所需的图元。

例如，当前有一个客户地点图层和一个人口普查地点图层。如果图层在地图中的排序不正确，应用程序将先绘制客户地点图层，然后在绘制人口普查地点图层。客户地点将由于人口普查地点图层的存在而变得模糊。

要更改图层显示的顺序，可以使用上移、下移按钮。选择要重新排序的图层，然后单击上移或下移按钮，将图层移至当前位置之上或之下的新位置。要将专题重新排序，必须对关联的基础图层重新排序。如果该基础图层有多个专题，则可在专题列表中重新对专题排序。

图层对象排序

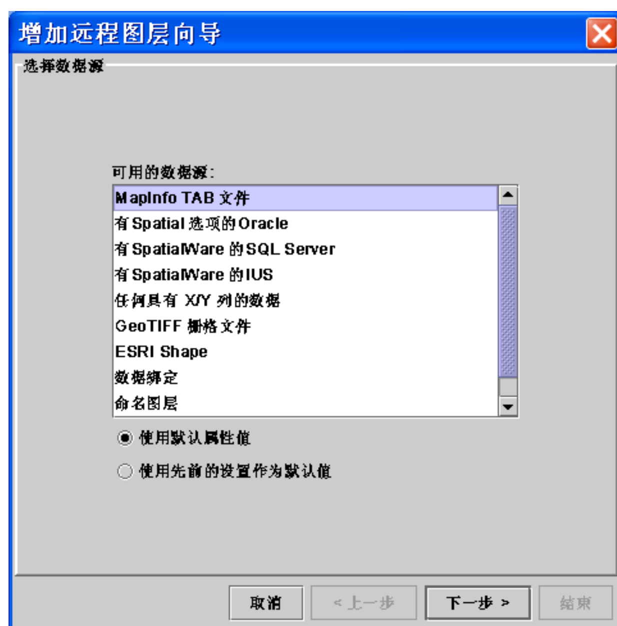
MapXtreme 不允许在单一地图图层中控制对象的前后排序。如果需要控制此类排序 （如需要确保某些线条显示在区域之上），可在各个图层中放置不同的对象类型。将线对象置于一个图层之中，然后将区域对象置于另一个图层之中。然后使用 “图层控制”对话框来对图层排序。

添加图层：增加图层向导

图层控制提供了增加图层向导来令添加图层的操作更加轻松。图层可以从多种来源来添加，其中包括：

- 文件
- 远程数据库
- 此前保存的命名图层
- 数据边界图层 （两个来源）

要向地图添加图层，可单击 “添加”按钮，然后遵循系统提示 （有关说明请参阅第 70 页的*使用增加图层向导*）。



数据源信息源自向导初始化的属性文件。数据源列表可以通过 **addlayerwizard.properties** 文件中的设置更改。此文件还包括用于向导中特定控件的默认值。其中还保存了用户为特定数据源输入的值，提供了下次运行增加图层向导时使用这些值的选项。

文件 addlayerwizard.properties 位于 MapXtreme-4.70/lib/client 之下。对其作出所需修改，即可更改增加图层向导的配置来满足具体的需求。上述操作假定 MapXtreme Java 管理器客户机正在作为独立的应用程序运行。

addlayerwizard.properties 的另一个副本存储在位于 webapps/mapxtreme47/client 的 mjmappletsup.jar 文件之内，该副本在 MapXtreme Java 管理器作为 applet 运行时使用。当作为 applet 运行 MapXtreme Java 管理器时，将会使用略有不同的安全设置，因此其需要自己的属性文件副本。

如果需要编辑 addlayerwizard.properties 文件，确保编辑要计划运行的应用程序的属性文件副本。

有关自定义 addlayerwizard.properties 文件的详细信息，请参阅*第 17 章：自定义 AddLayer 向导*。

编辑图层：编辑图层向导

单击 “图层控制” 对话框中的 “编辑” 按钮即可显示编辑图层向导。使用该向导可查看和 / 或编辑图层属性，例如 JDBC 连接信息、坐标系和几何设置以及图层名称。“图层控制” 对话框中并未显示图层类型，因此如要查看相应类型，可使用编辑图层向导。使用编辑图层向导，既可作出微小改动，例如将显示 “图层控制” 对话框中的图层名称；也可作出较大的改动，例如远程数据源的主机名和端口。除注释图层 （Annotation 图层）之外的所有其他图层类型均可编辑。



编辑图层向导中的字段已置入相应信息，除此之外，其外观和操作与增加图层向导类似。除了图层的类型之外，可更改图层的任意其他属性。例如，不能将 TAB 图层重定义为 JDBC 图层。

在随向导操作完毕之后，所选图层将采用新的属性构建，并将取代此前所选的图层。

注：　编辑图层向导有时将忽略图层专题和定制标注表达式。如果是使用该向导完全重新定义图层，则可能会出现这一情况，同时导致所生成的表和原始的表具有不同的列。

如果关注可能缺失的专题，可在使用过编辑图层向导之后在 “图层控制” 对话框中进行检查。如果专题处于维护之中，则将会看到相应专题列出 （可能需要单击图层的展开 / 压缩图标以进行显示）。如果这些专题并未列出，但是您需要保留相应专题，可单击图层控制的 “取消” 按钮，以取消在编辑图层向导中所做的更改。

移除图层和专题

要移除图层，可选择要移除的图层，然后单击 “移除” 按钮。所选的图层将从图层列表中移除。在 “图层控制” 对话框中，单击 “确定” 或 “应用”，重新显示该地图。这些图层并未保存为地图定义。

注：　移除操作并未删除任何文件。只是防止图层显示在这一特定地图中。

专题按钮

如果已经创建图层专题或标注专题，则可以通过图层控制的 “专题” 按钮控制缩放范围。此外，在此还可以更改距离单位。（使用地图定义界面上的添加专题工具即可创建专题。）

保存按钮

要将图层保存为命名图层，可使用图层控制的 “保存” 按钮。此时将转至保存命名图层对话框，在其中可指定图层名称及其存储位置。

“显示” 按钮

“显示” 按钮用于打开 “显示选项” 对话框，在其中可以自定义每个图层在地图中的显示，包括设置缩放范围和覆盖样式。

标注按钮

“标注选项” 对话框可通过图层控制的 “标注” 按钮打开，在该对话框中可创建多种标注样式用于地图图层。有关标注选项的说明，可参阅第 88 页的*通过图层控制显示选项*。

通过图层控制显示选项

在 “显示选项” 对话框中，可以自定义地图中每个图层的显示，包括设置图层缩放范围和覆盖样式。在 “图层控制” 对话框中，选择图层然后单击 “显示” 按钮，打开显示选项对话框。

缩放范围

在此对话框中设置图层显示的缩放级别。此外还可以设置用于图层的单位。缩放图层控制图层的显示，以便只在地图的缩放级别处于预设距离之内时显示。



例如，假设有一个街道图层和一个邮政编码边界图层。当缩小到 30 公里左右的距离时，街道的外观将会消失。这是因为缩放 （窗口宽度）太宽，无法显示街道地图的细节。

借助于缩放图层， MapXtreme 即可只在缩放设置为允许正确看到街道细节的距离的情况下显示街道图层，例如设置为小于 8 公里的距离。

同一地图中的不同图层可以显示在不同的缩放级别上。例如，假设有街道图层、县界图层和州界图层。我们希望只有在缩放级别小于 8 英里时令街道图层可见。在缩放级别处于 20 英里和 200 英里之间时显示县界图层。且只有在缩放级别大于 100 英里时令州界图层可见。

样式覆盖

使用不同样式来覆盖图层的显示是可用的显示选项之一，例如采用新的颜色用于区域、新的符号用于点、新的线形用于线段或区域边界等样式。这些样式覆盖均在 “显示选项” 对话框中设置。

可以使用 MapXtreme 附带的预定义样式来覆盖相应的显示。或者，可以采用多种方式来自定义这些命名样式，为地图图元提供所需的确切外观。将地图保存为地图定义，即可保存相应的样式更改。

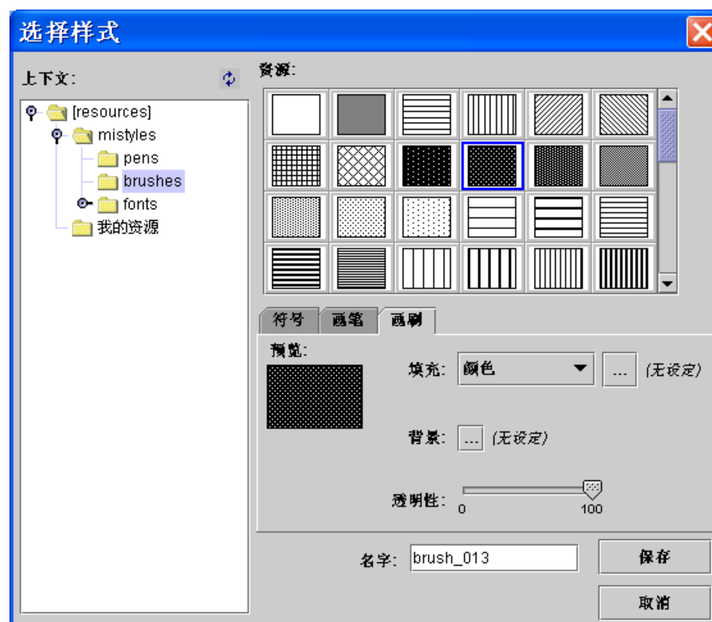
设置样式覆盖

要更改地图图元的样式，可在 “图层控制” 对话框中突出显示图层，然后单击 “显示” 按钮。在 “显示选项” 对话框中，单击 “选择样式” 按钮以打开 “选择样式” 对话框。单击左窗格中的 mistyles 上下文，显示可用样式的文件夹，然后选择适当的文件夹：画刷、符号或画笔。

使用画刷样式来更改地图图元的图案、填充或前景色。选择 Brush_002 可选取实填充，选择 Brushes_003-034 可用于图案。要设置透明填充，可选择 Brush_001。画刷编号在光标悬停在样式样本上时将显示工具提示。

使用字体样式可覆盖符号样式，其中包括符号类型、颜色、大小和单位。此处的单位可以是类似毫米的地图单位，也可以是类似英里的地理单位；借助于此，符号可在不同缩放级别上重新自行调整符号的大小。随 MapXtreme Java 安装了 9 种字体集。

使用画笔样式可更改线形、颜色、道路宽度或其他线形图元。画笔样式还可用于更改多边形图元的边界。



使用预定义的命名样式覆盖

要使用预定义的命名样式覆盖图层的显示样式，可从样板面板选择相应样式。样式将在名称框中保持其预定义的名称。单击 “加载” 以返回到 “显示选项” 对话框。点击 “确定” 两次，查看地图上的样式覆盖效果。

例如，要将所有星型更改为三角形，可从地图符号字体集选择三角形符号。单击“加载”，然后点击“确定”两次，即可查看地图。此时该符号将以其原始的磅值大小显示为三角形。注意此处的更改仅限于符号。符号仍然使用相同的磅值大小显示，因为覆盖的并非磅值大小（否则将视为是定制样式更改，有关说明如下所示）。

使用定制样式覆盖

使用定制样式覆盖图层的显示样式，可先选择预定义的样式，然后作出所需更改。对于颜色、大小和单位以及线宽所作更改将预定义的样式更改为定制样式。

例如，要将邮政编码边界从画刷样本，更改为带有粗框和蓝色阴影的图元，可选择所需的阴影图案。（如果此时加载这一图案，则该图案仍将视为预定义的命名样式）。单击填充颜色框，然后从颜色拾取器中选择蓝色。名称框中的样式名称将从 brush_001 更改为“定制样式”。在线宽框中，指定要应用于边框的新宽度。单击“加载”，然后点击“确定”两次，即可查看样式更改。

用于带有多种图元类型的图层的样式覆盖

同一图层中有多个图元类型也可设置样式覆盖。例如，要更改邮政编码边界的颜色和用于显示同一图层中的邮局符号的颜色，可先设置用于边界的样式覆盖。选择画刷图案和颜色。

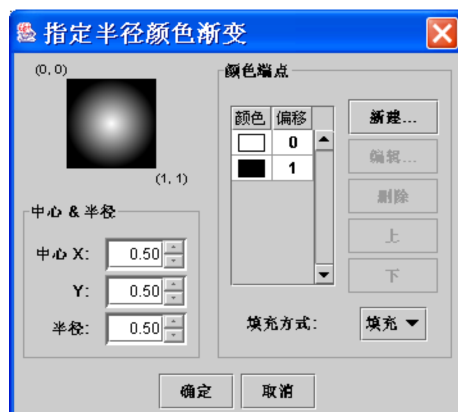
要更改符号样式的颜色，可单击“符号”选项卡来访问符号颜色选择器。选择新的颜色。注意所选样式窗口中的符号样式显示了采用新颜色的方形符号。这是表示颜色更改的默认符号样式。本例中该符号样式不会更改。将两种图元类型更改完毕之后，单击“加载”，然后单击“确定”两次，以查看在地图上所做更改。

使用渐变填充或单笔填充

可设置样式来使用渐变填充或单笔填充。可选使用线性或半径颜色渐变。线性渐变沿直线通过一系列颜色过渡。半径颜色渐变沿圆通过一系列颜色过渡。有关渐变的详细信息，请参阅第 260 页的*渐变*。

要使用渐变填充或单笔填充，可在选择样式对话框中选择半径颜色渐变或线性颜色渐变用于所要覆盖的填充或单笔属性。此后可以单击省略号按钮，显示“指定半径颜色渐变”或“指定线性颜色渐变”对话框。





此时即可指定满足特定需求的渐变特性。在对指定渐变满意之后，单击 “确定”。

保存新的命名样式

MapXtreme 附带有大量分类的标准命名样式，例如 brush_002 和 pen_118。此外，用户还可以自行创建定制命名样式。例如，可创建名为 BigBlueStar 的命名样式，表示 24 磅的蓝色星型符号。

要保存新的命名样式资源，可执行以下操作：

1. 运行 MapXtreme Java 管理器客户机。
2. 转至 “命名资源”选项卡。
3. 单击创建新样式按钮。此时将显示 “选择样式”对话框。
4. 选择类似 brush_002 的现有命名样式之一作为定制样式的起点。
5. 应用所需的定制样式属性 （颜色等）。
6. 在左侧的上下文树形结构中，单击用于保存新的命名样式的文件夹。
7. 在 “名称”文本框中，键入要用于新样式的名称。在此最好使用简洁明了的名称，例如 bluestar。
8. 单击保存按钮。此时新的命名样式即保存到服务器上。

注： 在从 “命名资源”选项卡启动之后，选择样式对话框只提供了 “保存”按钮。如果从类似 “图层控制”对话框的其他位置启动选择样式对话框，则所选样式将只能应用于地图，而不能保存到命名样式资源。 （这一限制是出于安全原因而设置的。如果 “选择样式”对话框始终允许用户保存命名样式，则任意使用图层控制 bean 的应用程序或 applet 都将允许任意用户创建或覆盖服务器上的命名样式）。

经由图层控制的标注选项

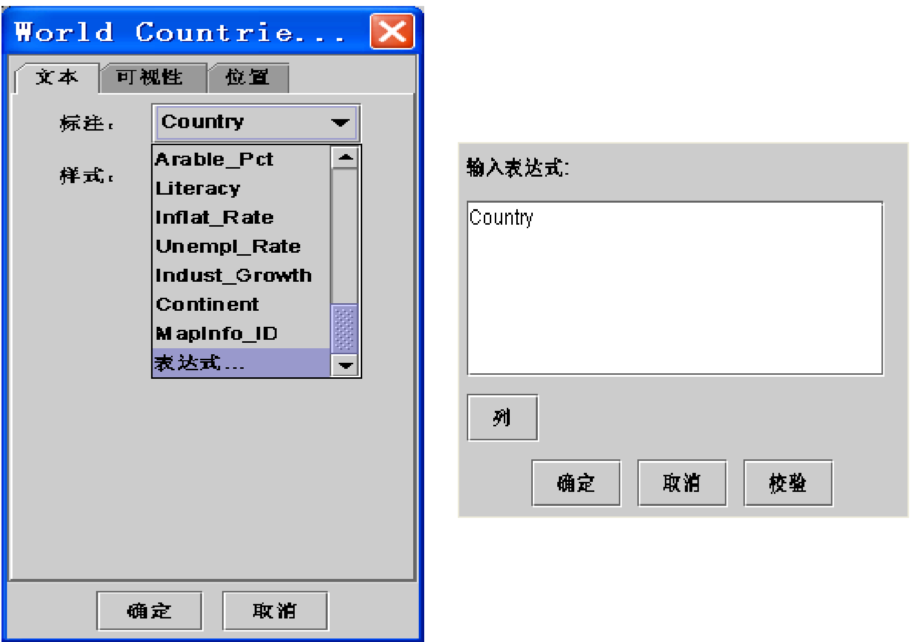
MapXtreme Java 提供了多种标注属性，用于在地图上实现精确复杂的明了外观。要更改图层的标注属性，可选择图层，然后单击 “标注” 按钮以显示标注选项对话框。

标注选项对话框中有三个面板，用于启用不同的标注特性：

- 文本 （列名或表达式、多行显示、字体属性）
- 可视性 （缩放设置、重复 / 重叠标注
- 位置 （自锚点的偏移量，水平和垂直对齐）

“标注文本”选项卡

要从 “文本” 选项卡设置用于标注的列，可从 “标注” 下拉列表中选择列。表的第一列为此列表的默认值。



此外，还可以使用列数据、静态文本或这两者的组合来创建表达式。要组合列数据和静态文本，将需要为该标注创建一个表达式。

例如，要创建一个标准，其起始为静态文本 "Pop:"，接下来是源自名为 POP_2000 的列的实际人口值。从样式下拉列表中，选择 “表达式”。此时将显示 “表达式”对话框。

在 “表达式”窗口键入 "Pop:"。确定该文件位于引号之内。从列的列表中选择 POP_2000 列。该列将自动输入到表达式窗口中。注意在静态文本和列输入之间将插入 + " " +。单击 “确定”，离开表达式对话框。单击 “确定”，离开图层控制，以查看标注表达式的结果。

如果要创建用于标注的表达式，并且希望在多行中显示该表达式，可在表达式窗口中，于表达式的换行处添加字符 "\n"。

例如：

```
Pop_2000 + " " + \n + Pop_Grw_Rate
```

将显示两行文本，说明每个特性的人口和增长率。

标注样式

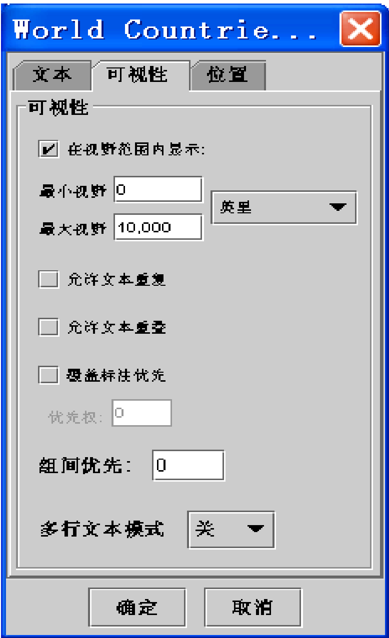
要更改标注的样式，可单击样式框来显示 “字体”选项对话框。从样式分类各种选项中进行选择，其中包括：字体、尺寸、像素点、颜色、背景 （光晕或方框）、轮廓、黑体、下划线和斜体。



单击 “选择颜色”框，打开颜色拾取器。此时既可从样本拾取颜色，也可在 RGB （红、绿、蓝）或 HSB （色调、饱和度、亮度）选项卡指定所需的确切颜色值。注意拾取颜色样本时， RGB 和 HSB 值将在其各自的选项卡中更新。

标注可视性选项卡

标注可以配置为只在特定的缩放范围之中显示，即采用在特定缩放范围中显示图层的相同方式。要指定标注的缩放范围，可选中 “在视野范围内显示” 复选框。随后可设置标注的最大和最小视野。



要在标注所有图层时避免地图混杂无序，可将标注设置为采用其他值缩放，以便在缩放时显示适当的标注。例如，将世界地图图层的缩放范围设置为最小视野为 0 英里，最大视野为 2000 英里。设置首都城市图层，以便在 200 和 500 英里之间显示标准。设置该城市图层，只在放大到 50 英里之下的范围中时显示标注。

选中或清除控制是否允许重复或重叠标签的复选框。如果需要使用同一标注在多个位置标注地图对象，则可选中 “允许文本重复”。如果不允许标注重复或重叠， MapXtreme Java 将只标注没有违反此设置的对象。切记如果地图上的重复或重叠标注过多，将令地图难于辨识。

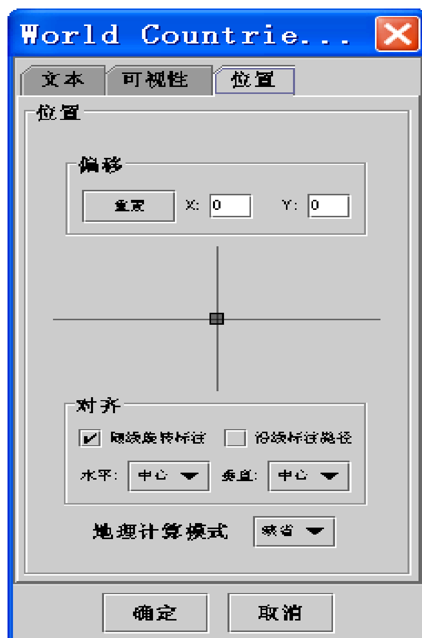
多个图层的标注将根据优先级和是否设置了重复和重叠文本来执行。最后绘制的图层比最先绘制的图层具有更高的优先级。要覆盖此优先级，可选中 “覆盖标注优先” 复选框，然后为该图层提供新的优先级编号。编号越高，优先级就越高。标注的默认优先级将通过（图层数量 － 图层排位）* 10。以上说明显示了世界地图的优先级为 30，因为其是 4 个图层之一并列在第 2 位 （图层位置 = 1）（位置 0 是最顶层）。

组间优先用于绑定标注之间的标注优先级。

要设置标注在多行显示，可将多行文本模式设置为开。标注文本中现有的换行仍将保留。关 （默认）表示所有文本均将显示为一行。计算表示 MapXtreme Java 将在进程中确定相应文本是否需要多行显示。计算操作是三个操作中速度最慢的操作，因为其要占用较多的运行时开支。

“标注位置”选项卡

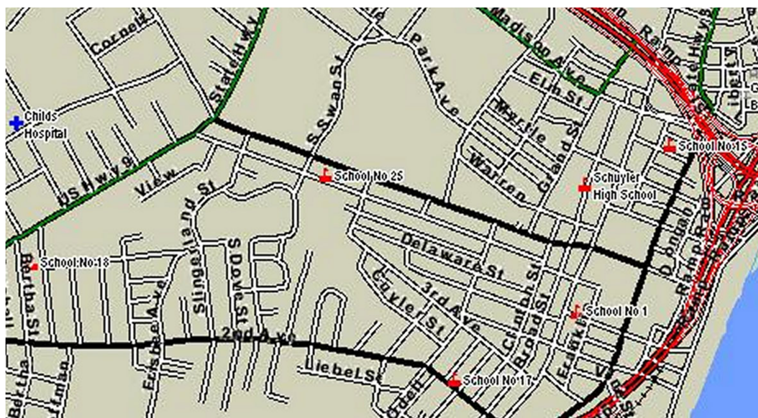
要控制标注位置，可在 “位置” 选项卡中设置偏移量距离，该距离是标注距离地图对象的标注点的距离。在 X 和 Y 框中，输入要用于新的标注位置的屏幕坐标，该坐标采用像素为单位。



此外，还可以设置用于该标注的水平和垂直对齐。标注可以在水平居于地图中心的左、中或右位置，或者可以垂直于基线，位于地图中心、顶部或底部。对齐特指最靠近标注点的标注边框的边。左对齐表示标注边框的左边缘最靠近标注点，即标注显示在右侧。

此外，还可以设置标注绕直线旋转。在 “标注选项”对话框中，该迷你面板将会更新以显示新设置相对于锚点的位置。

除了直线型的标注之外，还可以设置用于折线或多边形的标注，以遵循对象的曲线路径。选择此选项之后，标注即可沿折线或多边形的边界绘制。切记曲线标注基于所有标注设置在进程之中计算的，例如水平对齐、创意效果和可视性等标注设置。



某些设置不是用于曲线标注。多行文本模式目前尚不支持曲线标注。由于曲线标注的布放位置始终是动态计算的，因此将忽略地理计算模式。

地理计算模式可用于在更改地图视图时控制标注的布放。例如，如果放大到非常接近的距离，则设置为对象近似中心的标注点将有可能位于视野之外。要解决这个问题，可将计算模式设置为计算，以便 MapXtreme Java 在进程中于新的位置重绘标注。相应位置是根据剪切视图矩形来计算的。默认的行为 （静态）是不对标注点进行重新计算。原始标注点将用于每个视图。

管理命名资源

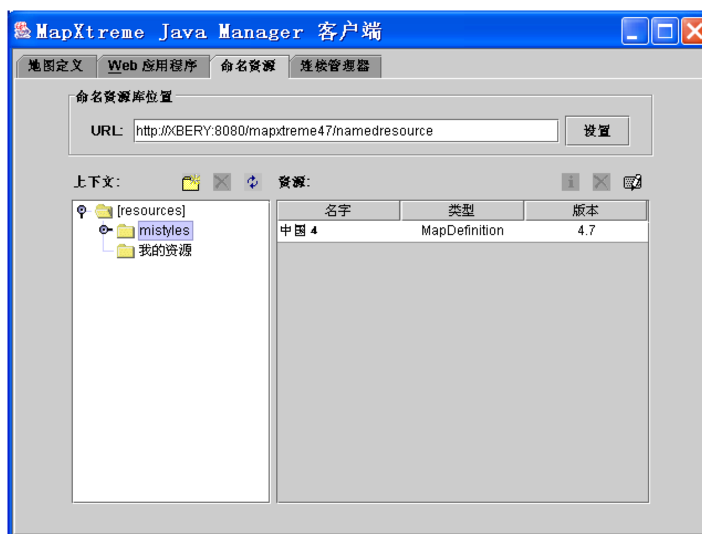
对于命名资源而言，一个资源将会附带一个名称。这提供了如下所示的众多优点：

- 允许资源以其名称所知，而非其属性。
- 允许资源位于一点即可从众多位置引用，资源管理更轻松。
- 要更改应用程序或数据的外观或行为，只需更改资源，而非每个应用程序或数据文件。

MapXtreme Java 支持若干种可以更加轻松地存储和检索地图信息的命名资源。 MapXtreme Java 管理器提供了命名资源面板用于管理命名地图、命名图层和命名样式。连接管理器是用于命名数据库连接的管理工具。后续内容中介绍了命名资源面板、命名地图、图层和样式。在*第 12 章：访问远程数据*中，也提供了命名数据库连接和连接管理器面板的说明。

命名资源面板

MapXtreme Java 管理器上的命名资源面板可用于管理所有可支持的命名资源。下图显示了该面板，其中带有用于三个命名资源的上下文。 mistyles 上下文在安装 MapXtreme Java 之后自动可用，其中包括了用于填充、直线和符号的多种预定义样式。命名图层和命名地图的上下文是由用户创建的。如果要将任何图层或地图保存为命名资源，必须先在此创建其上下文。



面板顶部 “命名资源库位置”默认为 MapXtremeServlet 的 URL。如果在其他位置创建资源库，则可以在此设置路径，以便显示相应位置的命名资源。该位置可以是基于文件的 URL （如 file:///c:/mapdata/mymaps）或指向正在运行的 NamedResourcesServlet 的 HTTP URL。

上下文面板提供了用于创建和删除上下文的按钮。资源面板显示了用于突出显示的上下文的命名资源。要查看命名资源的属性，卡单击 "信息" 按钮。要删除资源，可单击 "删除" 按钮。

注：　mistyles 上下文中的资源为只读，不可删除。

命名地图

命名地图是通过别名表示的唯一图层集合，例如采用别名 *myeurope* 或 *salesterr*。用于定义命名地图的信息存储为 XML 地图定义。命名地图既可在 MapXtreme Java 管理器中创建，也可通过编程创建，如下所示（请参阅 *第 9 章：MapJ API*）。

创建命名地图

要使用 MapXtreme Java 管理器创建命名地图，可执行以下操作：

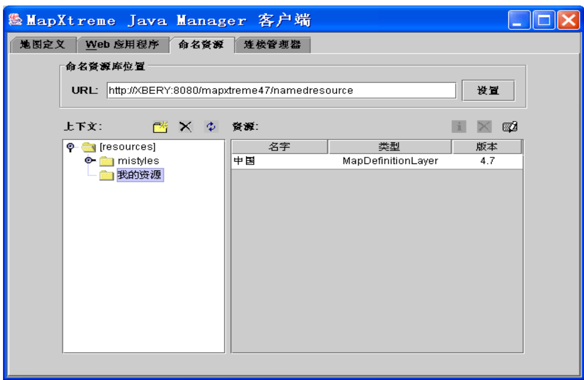
1. 从 MapXtreme Java 管理器客户端的地图定义面板，打开现有地图定义或 geoset 来显示地图。此外，还可以通过单击图层控制按钮并添加图层来构建地图。
2. 要对地图的显示作出调整，例如缩放、标注、添加专题图层等，均可使用在 MapXtreme Java 管理器中提供的工具来完成。
3. 单击保存按钮，将该地图保存为命名地图。此时将显示 "保存地图定义" 对话框。如果该对话框没有出现，可单击命名选项卡。



4. 单击位于左侧，用于存储命名地图的根上下文之下的子文件夹。
   如果没有用于存储命名地图的上下文，必须在保存命名地图之前，先在 MapXtreme Java 管理器的命名资源面板中创建相应的上下文。请参阅第 97 页的*命名资源面板* 。
5. 在 “名字”文本字段中，键入地图名称。单击保存。

命名图层

命名图层是已经指定唯一名称的图层，可通过相应名称在将来调用该图层。要将图层保存为命名图层，可单击 “图层控制”对话框中的 “保存”按钮，然后在 “保存命名图层”对话框中为其提供名称。

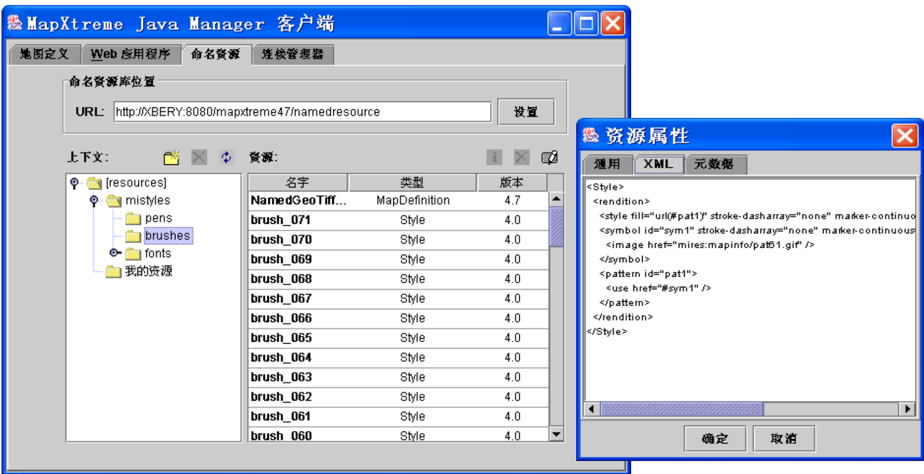


要从 “图层控制”对话框检索命名图层，可单击 “添加”按钮，以显示 “增加图层向导”。从提供的数据源列表中选择命名图层，然后遵循向导中的相关提示。

命名样式

MapXtreme Java 提供了众多填充、线形和符号样式，从中可选择用于覆盖图层的现有样式。单击 “图层控制”对话框中的 “显示”按钮，即可访问这些样式。有关 “显示选项”对话框的详细信息，请参阅第 88 页的*通过图层控制显示选项* 。

用于绘制直线和边界、填充区域和表示点位置的画笔）画刷和符号样式均为 XML 文件，这些文件默认情况下保存在 mapxtreme47/resources/mistyles 目录中。



这些样式和在 MapInfo Professional/MapBasic 中使用的标准样式相对应。要查看这些样式的缩略图，请参阅第 367 页附录 B 的*理解 MapBasic 样式字符串*。

通过更改颜色、线宽和符号大小，即可自定义预定义的样式，具体取决于样式的元素。要保存这些定制样式，必须保存地图定义，该操作将编写 XML 代码用于定制样式。定制样式不能在命名资源库中保存为新的命名样式，以便在其他地图定义中复用。

要通过编程保存定制样式，请参阅 MapJ API 和*第 14 章：标注和样式*以获取详细信息。

命名数据库连接

命名数据库连接只连接到通过别名引用的远程数据库，而不是通过例如驱动程序、 URL、用户名和口令等繁杂的 JDBC 详细信息引用的数据库。这些命名连接将存储在 miconnections.properties 文件中，该文件创建了一个 MapXtremeServlet 或 MapJ 可以访问的连接池。

命名数据库连接由 MapXtreme Java 管理器的连接管理器面板管理。有关详细信息，请参阅*第 12 章：访问远程数据*。

6

Web 应用程序构建器

本章详细介绍 MapXtreme Java 的 Web 应用程序原型开发向导。

本章内容：

使用 Web 应用程序构建器

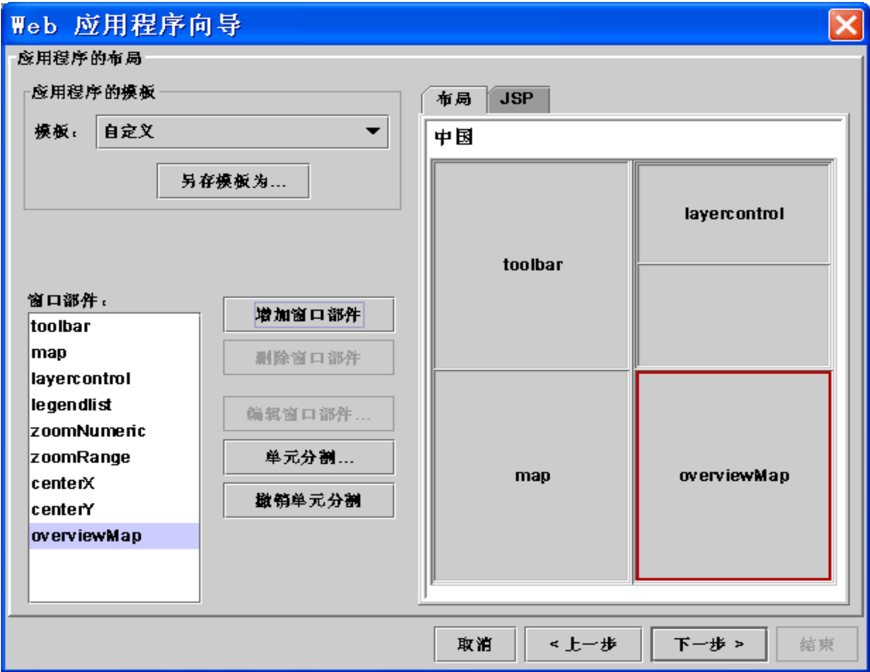
MapXtreme Java 管理器的 Web 应用程序构建器面板为开发人员提供了向导，指导开发人员轻松快捷地构建定制 Web 应用程序。借助于 Sun 的 Java 服务器页技术这一 Java 2 平台企业版的关键组件，开发人员使用这一快捷原型即可独立于平台，创建和维护 Web 页面中的静态和动态内容。 JSP 技术将用户界面和内容生成隔离开来，设计人员借助于此，更改整体页面布局时无需更改底层的动态内容。

这一原型构建器使用的标记类似于 JSP XML 定制标记，其行为类似于窗口部件，只需从列表中选择即可将其添加到构建器的布局框架中。我们在定制的 JSP 库中，提供了各种典型绘图窗口部件，例如工具栏、地图、图层控件和图例窗口部件 （将在第 106 页的*定制 JSP 标记库*中详细讨论）。有关构建定制 JSP 标记以及将其添加到向导的说明信息，请参阅*第 17 章：自定义 AddLayer 向导*。

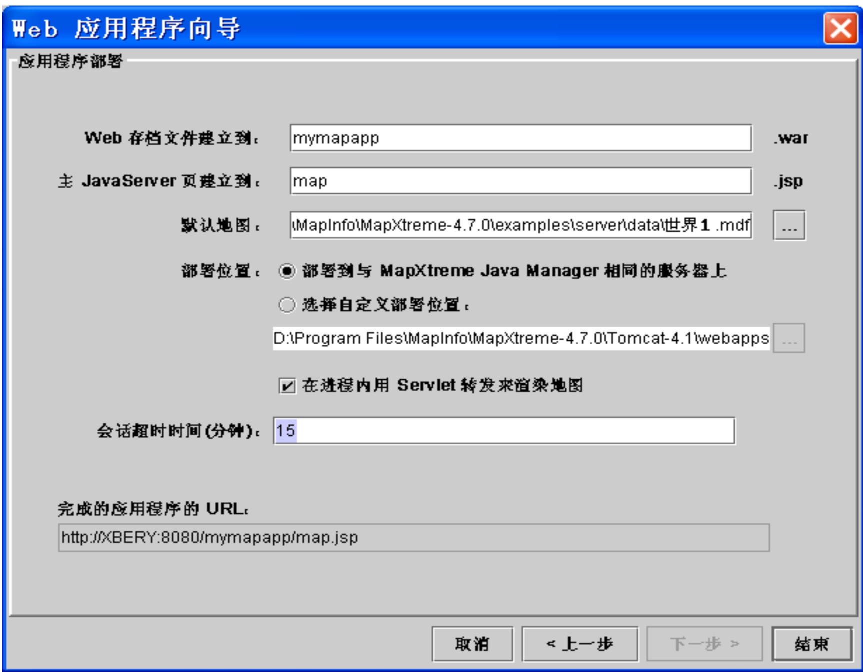
您所创建的应用程序将另存为 Web 存档文件 (.war) 中的 .JSP 文件。

Web 应用程序构建器向导将提供创建原型 Web 应用程序的逐个步骤说明。要开始创建，请执行以下操作：

1. 启动 MapXtreme Java 管理器客户端 （确保 MapXtreme Java 管理器 Servlet 正在运行）。
2. 单击 **WEB** 应用程序选项卡，显示 Web 应用程序构建器向导的第一个对话框。单击新建 **WEB** 应用程序按钮。为原型提供名称和说明，然后单击下一步。
3. 在“应用程序的布局”对话框中，单击对话框右侧的布局框架，以激活布局操作。



4. 从左侧列表框中选择窗口部件，然后单击增加窗口部件按钮。相应窗口部件将出现在 “布局” 框中。
5. 要添加其他窗口部件，可单击单元分割按钮，然后选择拆分布局框架的方式 — 水平或垂直。此外还可调整拆分单元的大小以满足具体的需求。
6. 在新单元处于激活状态时，突出显示窗口部件，然后单击增加窗口部件按钮。
7. 继续拆分单元，然后添加窗口部件，直至目标应用程序包含所需的全部元素。单击下一步。
8. 通过选择特定窗口部件，然后单击编辑窗口部件按钮；或者直接双击特定窗口部件，均可编辑布局之内任意窗口部件的属性。有关详细信息，请参阅第 105 页的*编辑窗口部件*。
9. “Web 应用程序向导” 对话框中提供了创建应用程序的有关信息，其中包括 Web 存档的名称 （*somefile*.war）、主 Java 服务器页 （*mymapapp*.jsp）以及关联的地图文件 （单击 **...** 按钮，浏览至具体位置）。



注：　要更改部署目录，先选择选择定制部署位置选项。

10. 如果部署到与 MapXtreme Java 管理器 servlet 相同的容器之内，可以选择启用 Servlet 转发，采用这种优化方式来渲染地图。有关这一特性的详细信息，请参阅第 213 页的*代码示例 :Servlet 转发* 。
11. 单击完成以创建应用程序。
12. 要对应用程序进行测试，可重新启动 servlet 容器，令新的应用程序可用。在 Web 浏览器中，键入在部署对话框中列出的 URL。

注：　如果使用的是 Tomcat 并且在默认位置创建 .war 文件，部署 .war 文件只需重新启动 Tomcat 即可。对于其他应用程序服务器而言，在部署 .war 文件之前，可能需要采取其他附加的操作步骤。

使用原型构建器模板

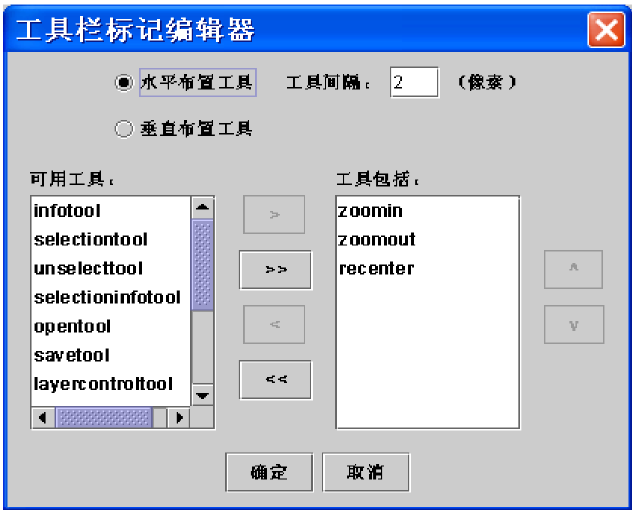
您可以保存所创建的任意布局，并在将来构建应用程序时将其用作新模板。对应用程序布局感到满意之后，单击另存模板为按钮，然后提供模板名称。在此后选择构建原型时，即可从下拉列表中选择模板名称，并在必要时对布局进行进一步的修改。

编辑窗口部件

Web 应用程序构建器的“应用程序布局”对话框包含编辑窗口部件按钮，可用于打开每个窗口部件的编辑器对话框。在该对话框中可控制多种属性，这些属性将影响到地图窗口部件的外观和行为。在向布局框架添加窗口部件之后，可单击编辑窗口部件按钮，以显示编辑器对话框。

例如，要从工具栏窗口部件对工具进行添加、删除或重新排序操作，请执行以下操作：

1. 单击“布局”框中的工具栏窗口部件，然后单击编辑窗口部件按钮或双击布局单元中的窗口部件。此时将显示“工具栏标记编辑器”对话框。



2. 在左侧窗格中突出显示特定工具，然后单击 [>] 按钮，即可将其添加到工具栏。
3. 要从工具栏删除工具，可在“要包含的工具”列表中将其突出显示，然后单击 [<] 按钮，将其从列表中删除。
4. 如有必要，可以继续添加或删除工具。使用对话框右侧的 [^] 和 [v] 按钮，即可根据需要对工具进行重新排序。
5. 如有必要，可将工具栏布局从水平更改为垂直，也可调整工具图标在工具栏上的间距。
6. 完成之后，单击确定。

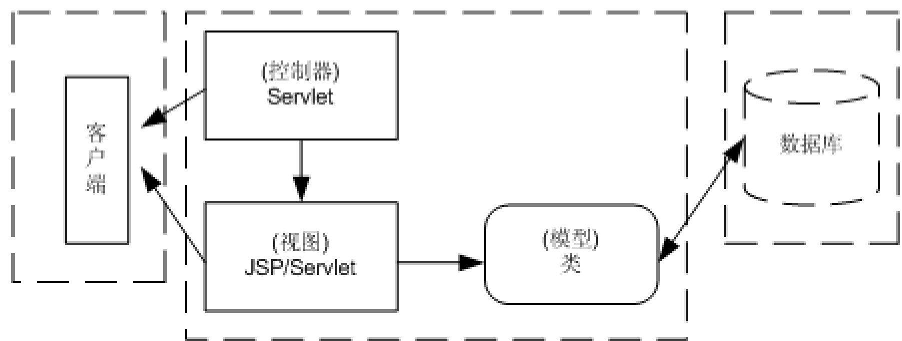
有关定制 JSP 标记库的后续内容提供了可以自定义的标记属性的详细信息。

使用原型构建器部署应用程序开发版本

如果要将应用程序部署到与 MapXtreme Java 管理器 servlet 相同的容器之内，可以选择启用 Servlet 转发，采用这种优化方式来渲染地图。相应操作可以通过 IntraServletContainerRenderer 来完成， MapXtremeServlet 借助于此，即可将其栅格数据直接写回到客户端。

定制 JSP 标记库

MapXtreme Java 提供了定制 JSP 标记库，使用文本编辑器或 IDE 即可将相应标记插入到 .JSP 文件中。这些标记在 MapXtreme Java 管理器的 Web 应用程序构建器中显示为窗口部件；在 Web 应用程序构建器中可选择所需元素，并将其添加到将要另存为 .JSP 文件的布局框中。在运行时， .JSP 将与执行应用程序业务逻辑的 servlet 通信。如需更改应用程序的显示，可在 Web 应用程序构建器中轻松重排、添加或删除窗口部件，创建新的 .JSP，而不会影响到生成 servlet 内容的操作。



这些定制标记设计可用于 MVC （模型 / 视图 / 控制器） JSP-servlet 体系架构。所生成的 .JSP （视图）包含表单，相应表单将提交到一般 servlet （控制器）。控制器 servlet 重定向至执行创建专题、执行半径搜索等必要业务逻辑的适当 Java Bean （模型），然后将相应请求转发回 .JSP 文件，由该文件显示更新的地图。

有关随 MapXtreme Java 提供的定制 JSP 标记库的列表和说明，请参阅*附录 A: 定制 JSP 标记库* 。

使用 JSP 标记库

所有 MapXtreme 定制 JSP 标记均包含在一个名为 **mapapp** 的主标记之内，该标记嵌入到 <mapinfo:mapapp> </mapinfo:mapapp> 标记块内。

在使用 MapXtreme 定制 JSP 标记构建定制应用程序时，务必注意如下所示的客户端浏览器有关要求：

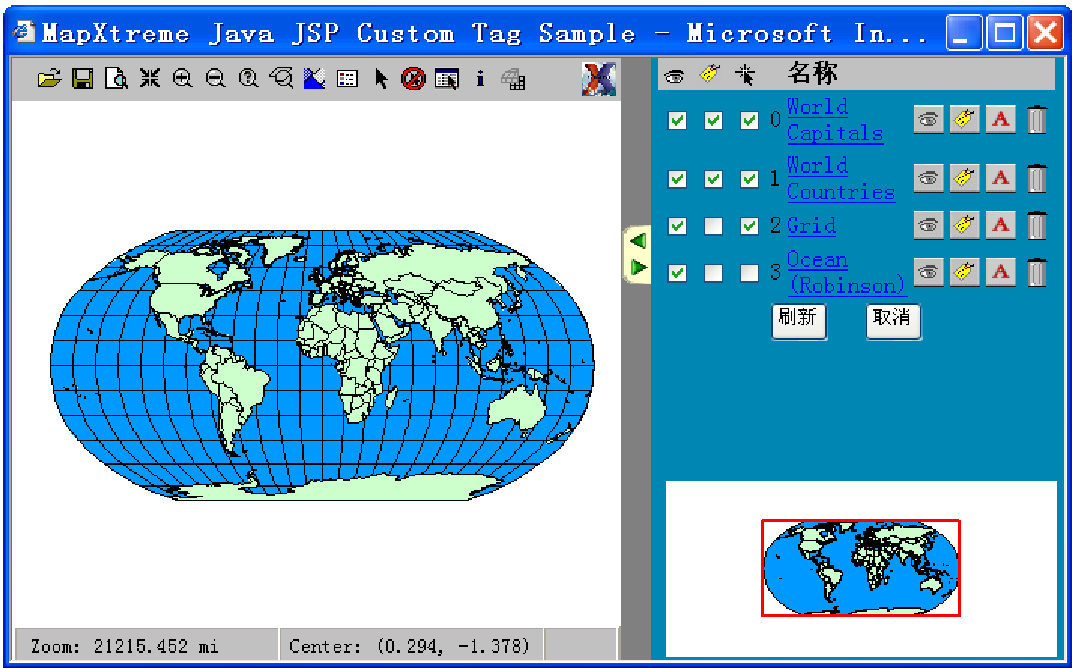
- JavaScript 必须启用 （不需要客户端 Java）。
- 浏览器版本相对较新 （例如 Netscape 4.7、 Internet Explorer 4.0 或更高版本）。

在服务器端，使用这些标记部署定制应用程序的 Web 服务器 / 应用服务器 /Web 应用服务器必须兼容 J2EE，即必须支持 JSP 1.1 API 和 Servlet 2.2 API。

要获取必要的初始化参数，可运行 Web 应用程序构建器向导在由该向导创建的 web.xml 中进行查找。

JSP 标记库示例：MapViewer

MapXtreme Java 提供了使用定制 JSP 标记构建的地图查看器应用程序作为示例。该应用程序即 mapviewer.war，在安装之后可见于 examples/client/jsp/mapviewer。这一示例可使用 JSP 库中可用的所有特性，并可显示自定义标记行为的具体方式。此外，这一示例还显示更高级的布局，相应布局不能使用 MapXtreme Java 管理器创建。



注： 此应用程序没有启用地理编码和线路规划特性。要启用相关特性，必须删除 web.xml 和 map.jsp 中的注释。此外，还必须令 MapMarker J 服务器和 Routing J 服务器运行相应特性。

7

MapXtreme JavaBeans

MapXtreme Java 版提供了 MapXtreme JavaBeans 帮助用户在 Java applet 或应用程序中轻松嵌入活动的向量地图。

本章内容：

概览

MapXtreme JavaBeans 利用最新的 Java 技术，并采用 Java 2 标准版平台中的 Java Swing 组件作为基础。在发布使用这些 JavaBeans 的 applet 时，如果最终用户的浏览器不支持 J2SE，那么将需要适当的插件。

MapXtreme JavaBeans 的自定义轻松便捷。即可构建在用户浏览器中运行的 applet，也可构建到独立的应用程序之中。 MapXtreme Java 附带的 MapXtreme Java 管理器就是一个使用本章所介绍的众多 JavaBeans 的应用程序示例。

MapXtreme JavaBeans 在类似 JDeveloper、JBuilder、Visual Age 的可视化开发环境中易于使用。创建地图绘制 applet 所需的组件可以轻松拖放到窗体之中。随后即可设置属性。

使用 MapXtreme JavaBeans 开发，不仅可以迅速创建地图绘制 applet ；而且与嵌入地图的常规 HTML 相比，还提供了更加强健的功能。这尤其适用于具备功能强大的服务器的内网环境，相应网络可以更好地处理由于下载 applet 所增加的下载时间。有关使用 applet 配置选项的详细信息，请参阅*第 3 章：应用程序规划*。

此外，使用 MapXtreme JavaBeans 还可以更加迅速地进行原型开发。

在 applet 或应用程序开发中，共有如下 7 种类型的 MapXtreme JavaBeans 可供使用 :

- VisualMapJ Bean
- MapToolBar Bean
- MapTool Beans
- LayerControl Bean
- AddTheme Wizard Bean
- LegendContainer Bean
- ViewEntireLayer Bean
- ZoomPanel Bean

后续内容中对上述各个 MapXtreme JavaBean 分别作出了说明。

VisualMapJ Bean

VisualMapJ Bean 是在内容窗格中显示地图的主要地图绘制组件。该 Bean 构建在现有 MapJ 类库的顶部。使用 VisualMapJ，可在可视化的开发环境中完成最常见的地图绘制操作，而无需通过编程。相应操作包括添加地图定义或 geoset、控制缩放、控制中心、控制边界等。此外，有些较为复杂的操作也可使用源自 API 底层的 MapJ 对象来完成。

VisualMapJ 基于由其内部 MapJ 引用维护的状态来显示地图图像。 VisualMapJ 维护了一个注册地图工具的列表，并管理工具的选择和取消选择。它将鼠标和键盘活动转发到当前所选地图工具，然后依靠相应工具来控制其状态。

VisualMapJ 既可和 MapXtremeServlet 协同工作，也可单独使用。 **StartupMapDefinition** 属性可用于指定初始化 VisualMapJ (MapJ) 的地图定义或 geoset 文件。在此文件中定义的图层指定数据访问是直接进行还是通过 MapXtremeServlet 来完成。在 applet 中使用 VisualMapJ 时，需要设置通过 MapXtremeServlet 来完成数据访问，以便符合由 applet 环境所强制的安全约束条件。

MapRenderer 属性用于指定地图是本地渲染 （在嵌入 VisualMapJ 的 applet 或应用程序的处理空间之内），还是通过 MapXtremeServlet 远程渲染。

ShowToolTips 属性指定当鼠标在 VisualMapJ 范围内悬停时，是否显示弹出式文本。这取决于设置弹出式文本内容的单独 MapTool。

最后， VisualMapJ 提供的 **Center** 和 **Zoom** 属性还可指定地图的初始界限。

MapTool Beans

MapTool Beans 是可用于执行基本地图导航和选择操作的工具集合。可实现包括平移、重新对中、缩放、测量距离，以及在给定点或限定边界内获取信息和选择图元在内的众多操作。下表提供了相应的工具列表。

所有 MapTool Beans 均扩展 Swing 的操作接口。这一点令其可以更加轻松地添加到工具栏或菜单。如果将其同时添加到工具栏和菜单，那么其间将会保持同步。

MapTools 在 VisualMapJ 上的工作领域如下所示：

- 指定要显示工具提示
- 指定要使用的光标
- 在地图图像顶部提供定制渲染 （如选取框、标尺弹性线）

- 修改任意 MapJ 属性 （中心、缩放、图层、专题）
- 提示 VisualMapJ 重画。

下表列示并说明了可用的 MapTools。

MapTool		目的
PanMapTool		用于通过拖放将地图在其窗口中重新定位。
RecenterMapTool		用于将地图中心重新定位到点击点。
RulerMapTool		用于获取两点或多点之间的距离。
InfoMapTool		用于获取点击点的属性信息。
ZoomInMapTool		用于获取更近的地图或图层区域视图。在用户拖动鼠标时，将出现黑色的小选取框。
ZoomInMapTool2		用于获取更近的地图或图层区域视图。在用户拖动鼠标时，将出现成斜角的选取框。
ZoomOutMapTool		用于获取地图的较宽视图。
ObjectSelectionMapTool		在鼠标单击位置选择单独的图元。
BoundarySelectionMapTool		选择包含在最顶部可见图层中唯一区域图元的边界多边形中的图元。
RadiusSelectionMapTool		选择包含在通过鼠标拖动操作形成的边界圆圈中的图元。
RectangleSelectionMapTool		选择包含在通过鼠标拖动操作形成的边界矩形中的图元。
PolygonSelectionMapTool		选择包含在通过一系列鼠标拖动操作形成的边界多边形中的图元。

选择 MapTools 允许用户通过单独的鼠标单击和通过更加复杂的鼠标拖动操作来选择多个图层中的图元。这些鼠标交互操作可在每个图层的基础上创建 SelectionThemes，表示使用 VisualMapJ 地图中的工具所做的选择。有关 SelectionThemes 的详细信息，请参阅*第 13 章：图元和搜索* 。

MapToolBar Bean

地图工具可添加到任意 Swing JToolbar。 MapXtreme JavaBeans 附带有一个特殊的 MapToolBar 组件。默认情况下， MapToolBar 包括 ZoomInMapTool2、 ZoomOutMapTool、 PanMapTool、 RulerMapTool 和 InfoMapTool。工具栏还包括一个用于激活 LayerControl Bean 的按钮。这些默认项目均可删除，并且也可以将其他附加工具添加到工具栏。

VisualMapJ 和 MapToolBar 具有相同的父级组件，并且两者都必须以特定顺序添加以便正确注册，这是使用 MapToolBar Bean 的要求之一。必须先将 VisualMapJ 对象添加到窗体，然后再添加 MapToolBar Bean。按此顺序添加组件时， MapToolBar Bean 将其所有 MapTools 注册到 VisualMapJ 组件。如果先添加的是 MapToolBar Bean，则还需要手动关联 MapTools 和 VisualMapJ。

在添加工具栏之后，即可通过单击其相应按钮，令一个工具成为当前所选的工具。

有关使用 MapTools 的详细信息，请参阅第 123 页。

LayerControl Bean

为赋予用户控制地图图层的功能，可将 Layer Control Bean 包含在应用程序或 applet 的 ToolBar 中。

Layer Control Bean 提供了一个用户界面，可用于设置图层可见性、可选性和自动标注，以及添加、编辑、删除或重排图层。此外，还可设置图层显示特征和标注属性以及管理专题。

如果在 GUI 中包含了默认的 MapToolBar，则可通过单击图层控制按钮来显示图层控制。



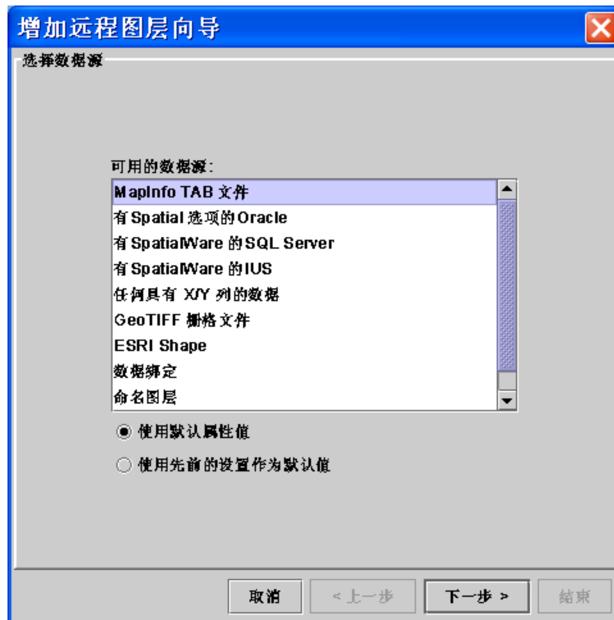
有关图层控制构成元素的完全说明，请参阅第 83 页的*图层控制* 。



AddLayer 向导

向地图添加图层是图层控制的主要功能之一。单击添加按钮显示增加图层向导，该向导将指导用户完成从数据源添加图层的各个步骤；其中的数据源既可是本地存储的 tab 文件，也可是经由 JDBC、命名图层或数据绑定图层访问的数据源。

注：　在 applet 中，不能添加某些图层类型，其中包括 .TAB 文件图层，原因在于 applet 的安全性阻止了 applet 访问本地文件系统。这一限制并不适用于应用程序。



增加图层向导可以配置，以便协助您为用户轻松添加图层提供帮助。配置信息存储在 /MapXtreme-4.7.0/lib/client 目录下的 **addlayerwizard.properties** 文件中。在该属性文件中，可定义用于公共设置（如 Oracle Spatial 主机和端口）的预设值。该向导将记忆每个数据提供方最后使用的值。通过属性文件，可从向导增减特定的数据提供方和命名资源。在文件中存储口令为可选（默认为不存储口令）。

有关 addlayerwizard.properties 文件的详细信息，请参阅*第 17 章：自定义 AddLayer 向导*中有关 addlayerproperties.file 的介绍。有关在 MapXtreme Java 管理器中执行增加图层向导的逐步说明，请参阅第 85 页的*添加图层：增加图层向导*。有关命名资源的详细信息，请参阅*第 12 章：访问远程数据*。

EditLayer 向导

将图层添加到地图后，即可通过 EditLayer 向导来查看或更改其定义属性。从图层控制，单击编辑按钮来显示该向导。继续在相应面板（与 AddLayer 向导中的面板相同）执行适当操作，以查看和 / 或作出任意更改。在单击完成之后，图层将使用新指定属性来重建。

AddTheme Bean

AddTheme Bean 是一个向导工具，用于帮助用户将图元专题或标记专题添加到地图。



图元专题

对于图元专题，可以创建范围主题，根据一段范围内的值来划分图元；也可以创建独立值专题，根据值来以影线表示图元。



图元主题可以基于任何支持的列和当前地图中的图层。目前支持基于数字、字符串和日期列数据创建专题，以及基于点、线和区域图层创建专题。

部分操作将用于创建与新专题相关联的默认专题图例，。图例可以自定义，以更改标题、字体、插入信息、说明文本和颜色。

通过向导 Bean，用户可以选择专题名称。对于 RangedThemes，用户还可以选择范围 (bin) 数量、分布方法（等计数或等范围），以及所有分断值（bin 的上限值）的舍入值。用户还可以指定将要通过起止范围样式的设置，根据专题来以影线表示的点、线或区域的样式（颜色）。随后将计算适当的样式，用于起止范围之间的对象。

对于 IndividualValueThemes，用户可以选择将图层中的哪些值进行有所区别的影线表示，以将其与其他值区分开来。

在运行时， AddTheme Bean 关联到现有 VisualMapJ 例程。为此， VisualMapJ 必须是 AddTheme Bean 所要添加到的组件的子级。例如，如果将 AddTheme Bean 添加到已添加 VisualMapJ 例程的 JPanel，则将会发现 VisualMapJ 例程并与之关联。如果无法进行自动关联，则必须使用 AddTheme Bean 的 setVisualMapJ(VisualMapJ) 方法来将 VisualMapJ 的工作示例置入 Bean 中。

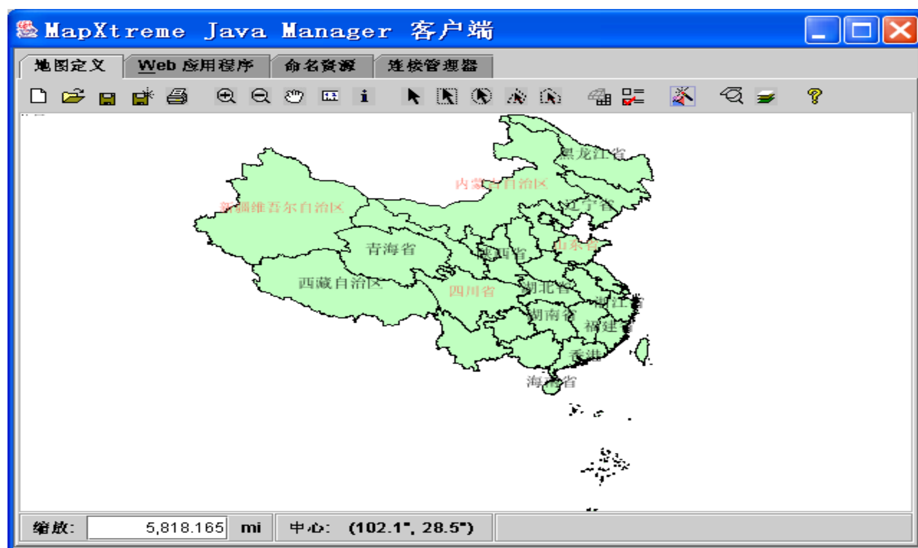
AddTheme Bean 扩展 AbstractAction，因此其可以添加到 JMenu 或 JToolbar （菜单和工具栏将保持同步）。在单击菜单项或工具栏之后，将会显示 AddTheme 向导。

有关通过编程来创建范围主题和独立值图元专题的详细信息，请参阅*第 15 章：专题地图绘制和分析* 。

标注专题

MapXtreme Java 支持 4 种类型的标注专题：范围、独立值、覆盖和选择。当要在同一图层中为标注采用不同属性时，可使用标注专题。

例如，假设正在使用人口增长率标注世界各国图层。对于增长率较高的国家，需要其比增长率较低的国家更加突出。通过 Add Theme Bean 创建范围标注专题，划分图层值的范围，将颜色和 / 或字体大小传递到标注即可实现上述目标。



有关标注专题的详细信息，请参阅*第 14 章：标注和样式* 。

LegendContainer Bean

作为管理图例布局和显示的一种方式， MapXtreme Java 包括了 LegendContainer Bean。LegendContainer Bean 可以拖至侧边的 VisualMapJ 对象中，然后显示和 VisualMapJ 中的图层关联的任意图例。如果 LegendContainer 检测到专题的添加或删除，则将对其显示作出相应更新。

图例是 Swing JPanels。因此，图例可布置在 LegendContainer 中，并自行渲染到 LegendContainer 中。

ViewEntireLayer Bean

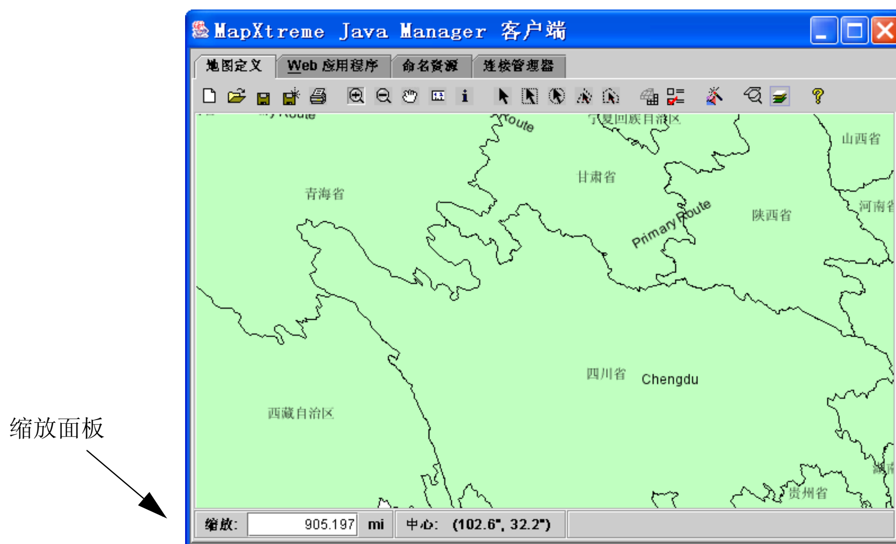
ViewEntireLayer Bean 可用于地图的迅速对中和缩放，以便充分缩放，看到图层全貌。ViewEntireLayer Bean 可以添加到 MapToolBar。单击 MapToolBar 上的查看整个图层按钮，可显示查看整个图层对话框。用户从中可以选择要查看的图层。



SimpleMap 示例 applet 提供了使用 ViewEntireLayer Bean 的示例。可见于 /examples/client/Java/simplemap。有关 SimpleMap applet 的详细信息，请参阅第 122 页的 *SimpleMap Applet 示例*。

ZoomPanel Bean

ZoomPanel Bean 提供了一个显示当前地图缩放的文本字段。可用于用户键入新的缩放宽度。



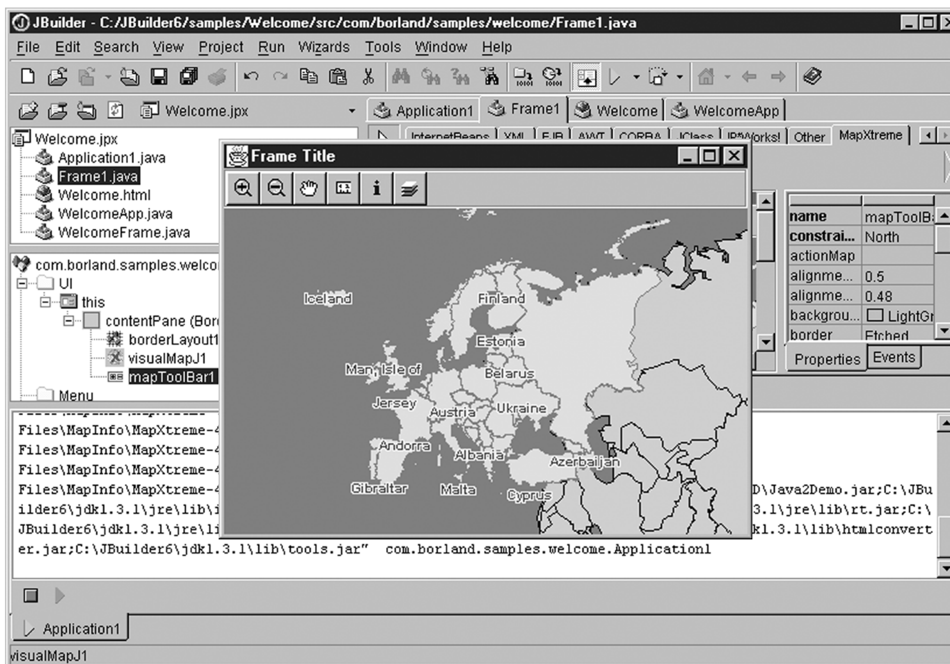
有关 ZoomPanel bean 的其他示例，请参阅 /examples 文件夹下的 SimpleMap 示例 applet。

使用 MapXtreme JavaBeans 创建应用程序

本节介绍使用 MapXtreme JavaBeans 创建独立地图绘制应用程序的步骤。

本例使用 JBuilder 6.0 个人版作为 RAD 环境。有关其他环境的详细信息，可使用本例作为指南，同时参阅文档以获取相关特定说明。

如果要采用可视化方式来绘制用户界面，即所见即所得的方式，则必须将 MapXtreme 的 Java Bean 集成到 JBuilder 的一个组件调色板中，如下所示。按照此方式，我们还将定义 MapXtreme 库 （一组 jar 文件）；即使不用所见即所得的方式来创建用户界面，这个库也非常实用。



将 MapXtreme Java Beans 添加到 JBuilder 6 IDE

1. 在 “文件”菜单中，单击新建，显示对象库。
2. 选择应用程序，然后单击确定。 JBuilder 将创建一个带有标准示例应用程序的新项目。
3. 单击设计选项卡，切换到 GUI 编辑模式。此时，右上部将出现一个归类的标签调色板 （Swing、 Swing 容器等）。
4. 右键单击 **SWING** 选项卡，然后单击属性。此时将出现调色板对话框。
5. 单击添加以定义新页，提供页面的名称，例如 MapXtreme。
6. 在对话框左侧的 “页面”列表中选择 MapXtreme 项目。
7. 单击添加组件选项卡。
8. 单击选择库按钮。此时将出现 “选择其他库”对话框。
9. 单击新建按钮。此时将出现新建库向导。
10. 键入用于新库的名称，例如 mapxtreme_lib。
11. 单击添加按钮，然后选择 MapXtreme's lib/client/mxjbeans.jar 文件。“库路径”列表现在应已包括 mxjbeans.jar。
12. 再次单击添加按钮，将目录添加到 “库路径”列表 － MapXtreme lib/client 目录。

    注： 务必只选择该目录。而不要展开该目录来显示其内容。

13. 再次单击添加按钮，将以下 .jar 文件添加到 “库路径”列表 （大多数情况下均位于 lib/common 目录中）：mxj.jar、 micsys.jar、 miutil.jar、 jdom.jar、 xercesImpl.jar、 mxjtabdp.jar、 mistyles.jar、 mxjloc.jar 和 xml-apis.jar。
14. 单击确定，关闭新建库向导。
15. 确保选定新的 mapxtreme_lib 库，然后单击确定，关闭 “选择其他库”对话框。
16. 在 “组件过滤”框中，单击仅限 **JAVABEANS** 选项。
17. 单击从所选库添加按钮。此时将出现 “浏览类”对话框，显示以 com 为起始的类树。
18. 展开 com 树形结构，直至看到 com.mapinfo.beans.vmapj 文件包的内容。选择 VisualMapJ 类，然后单击确定。此时将出现 “结果”对话框，确认已经将 VisualMapJ 类添加到调色板。
19. 重复第 16 步 和 第 17 步，将更多 Java Beans 添加到调色板，但这次选择的不是 com.mapinfo.beans.vmapj.VisualMapJ 类，而是 com.mapinfo.beans.tools.MapToolBar。
20. 单击确定，关闭 “调色板属性”对话框。
21. 在位于屏幕右上部的标签调色板区域，单击 MapXtreme 选项卡。（此时可能需要滚动显示才能查看 MapXtreme 选项卡，具体取决于屏幕大小。）在 MapXtreme 调色板中，可以看到对于每个已添加的 Java Beans，都会有一个相应的图标 － 本例中的一个图标用于 VisualMapJ，另一个用于 MapToolBar。

在此可向调色板添加更多的 MapXtreme Java Beans，但是本例出于简化的目的，将只使用 VisualMapJ bean 和 MapToolBar bean。

在将 MapXtreme Java Beans 添加到 JBuilder 之后，即可使用相应的 beans 来基于 MapXtreme Java beans 绘制 Swing 用户界面。

1. 在左侧较低的框中，单击标注为 contentPane (borderLayout) 的灰色矩形图标。
2. 单击 MapXtreme 组件调色板上的 VisualMapJ 图标。
3. 在表示应用程序的灰色矩形上绘制一个矩形。这一操作将 VisualMapJ 组件置于应用程序之上 （在 BorderLayout 的中点）。
4. 在左下侧选中了 VisualMapJ1 项之后，单击右侧的属性检查器面板中的 startupMapDef 属性的右侧。然后单击 **...** 按钮，打开 “开始 MapDef”对话框。
5. 单击更改起始地图定义按钮，然后选择 .mdf 文件，例如 world.mdf （位于 MapXtreme 的 examples/server/data 目录之下）。
6. 单击组件调色板上的 MapToolBar 图标。
7. 直接单击左下侧的框中 contentPane (borderLayout) 项。这一操作将 MapToolBar 置于 BorderLayout 的 “北方”点。
8. 在 “运行”菜单，单击运行项目。此时将会启动应用程序，可用于查看地图并使用 MapXtreme 的工具栏按钮与地图交互 （放大等）。

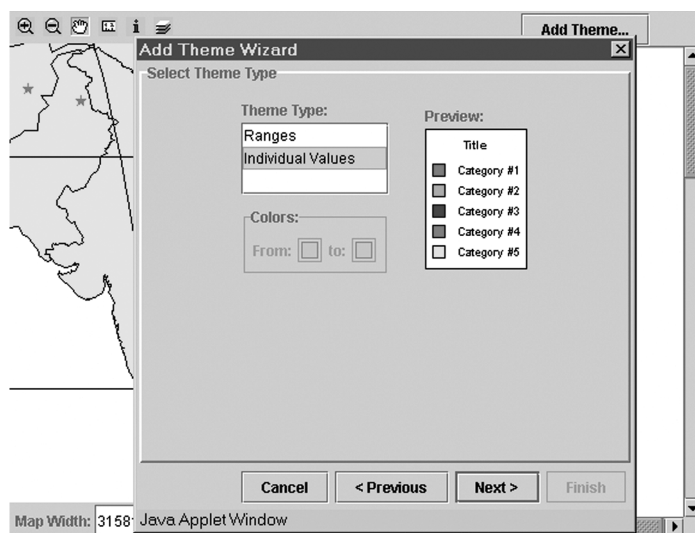
SimpleMap Applet 示例

随 MapXtreme Java 作为示例应用程序提供的是一个名为 SimpleMap 的 applet 示例，该示例使用 MapXtreme JavaBeans。此 applet 需要 MapXtreme 在 web 服务器上运行。

本节提供了该示例的概览。有关配置和运行 SimpleMap 的详细信息，请参阅位于 /examples/client/Java/simplemap 目录下的 readme.html 文件。

SimpleMap 是一个 applet，提供了以下地图查看基本功能：

- 显示地图 （使用 MapXtreme JavaBean、 VisualMapJ）。
- 显示包含用于地图导航的工具栏、信息工具和图层控制工具 （使用 MapToolbar Bean）。
- 允许用户通过单击 “添加专题”按钮来添加专题影线表示 （使用 AddTheme Bean）。
- 在地图右侧的面板中显示相应的专题图例 （使用 LegendContainer Bean）。
- 提供一个文本字段，显示当前地图的缩放，允许用户键入新的缩放宽度 （使用 ZoomPanel Bean）。



SimpleMap 示例包含以下文件：

- Readme.html － 说明 applet 以及其实施步骤的文档。
- SimpleMap.java － 主代码模块。由于使用了 MapXtreme Beans，因此其源代码非常简单 － 尤其是使用 VisualMapJ Bean、Map Toolbar Bean 和 AddTheme Bean － 来完成大部分工作。
- simplemap.html － 示例 HTML 文件，显示如何运行使用 Sun 的 Java 插件的 applet。这一 HTML 页不再使用 <APPLET> 标记，而是使用 Sun 的 Java 插件所使用的

<OBJECT> 和 <EMBED> 标记。如果没有安装 Java 插件，查看此 HTML 页时，系统将提示您下载相应插件。

SimpleMap Applet 安装

以下提供了安装 SimpleMap 所需步骤的概要。有关完整的详细信息，请参阅 SimpleMap 目录中的 readme.html。

1. 配置和启动用于运行 MapXtremeServlet 的 servlet。（*第 2 章：入门*提供了具体步骤。）
2. 复制以下文件：simplemap*.class 文件、simplemap.html、addlayerwizard.properties、resources-config.xml、 encoding-map.xml、地图定义文件，例如 world.mdf 和 MapXtreme jar 文件，拷贝到可通过 Web 浏览器访问的目录。
3. 如果已选择使用地图，而不是 world.mdf，那么修改 simplemap.html 中的标记，指定要使用的地图文件的名称。
4. 通过在浏览器中打开 HTML 文件来加载 applet。

切记运行 SimpleMap 必须正确配置以下项目：

- MapXtremeServlet 必须运行 （如 servlet 容器必须运行）。
- applet 必须正确编译。
- Web 浏览器必须安装 Java 插件。
- 用于加载 applet 的 HTML 文件必须指定适当的 param 标记。

Map 定义和 SimpleMap

如果配置 SimpleMap 来加载地图定义，务必确保该地图定义是使用 “增加图层向导”对话框中的由远程 **MAPXTREMESERVLET** 访问数据选项来构建的。如果尝试访问本地资源，则将会报告有关安全问题的错误消息。有关创建地图定义的详细信息，请参阅*第 5 章：管理 MapXtreme Java* 。

在 Applet 或应用程序中包含 MapTools

大部分地图绘制应用程序都允许用户通过鼠标动作与地图进行交互，执行各种任务 （如单击甚至拖动）。如果要支持此类地图交互，则需要向应用程序或 applet 添加地图工具。

如果开发环境支持可视化的 GUI 设计，还可采用可视化的方式创建和控制 GUI 元素，例如工具栏和工具。例如， JDeveloper 用户在设计模式中可以将 GUI 元素绘制到窗体中。如果开发环境不支持可视化的 GUI 构建，则需要手动键入如下所示代码创建地图工具。

创建 MapToolBar

借助于实例化 MapToolBar 对象，可轻松和迅速地向代码添加一组地图工具，然后将其添加到布局。默认的 MapToolBar 对象提供几种公共的地图工具，其中包括放大、缩小、平移、信息和标尺以及一个启动 “图层控制” 对话框的按钮。

实例化 MapToolBar 简便轻松，无需参数即可实现。

```
MapToolBar mapToolBar1 = new MapToolBar();
```

或者，如果要对这组包括工具栏在内的地图工具作出重要更改，还可使用采用地图工具向量的其他 MapToolBar 构造器。

在创建 MapToolBar 之后，将其添加到您已经将 VisualMapJ 对象添加到的相同容器中。

```
this.getContentPane().add(visualMapJ1, BorderLayout.CENTER);
this.getContentPane().add(mapToolBar1, BorderLayout.NORTH);
```

关联工具与地图

MapToolBar 中的工具必须与 VisualMapJ 对象相关联。如果没有正确关联，那么工具对于地图不起作用。

关联工具与 VisualMapJ 对象最简单的方式是将 MapToolBar 对象添加到 VisualMapJ 所在的相同容器 （如相同的 JPanel）。当 MapToolBar 和 VisualMapJ 对象位于相同的容器中时，工具将自动与该地图相关联。

如果地图工具没有和地图自动关联 （如因为 GUI 布局需要将工具栏置于其他 JPanel），则需要调用 VisualMapJ 对象的添加方法，将地图工具注册到地图，如下所示。

添加单独工具

如果发现随默认 MapToolBar 对象提供的工具无法满足所需地图工具的要求，则可自行添加更多工具。例如，默认的 MapToolBar object 对象不包括类似 RadiusSelectionMapTool 的选择工具，此时需要将该工具添加到工具栏。要将单独的工具添加到工具栏，可执行以下操作：

1. 创建计划使用的任意工具对象。
   ```
   RadiusSelectionMapTool radTool = new RadiusSelectionMapTool();
   ```
2. 如有必要，可调用 VisualMapJ 的添加方法，关联工具与地图。（如上所示，如果 MapToolBar 位于与 VisualMapJ 相同的容器中，即无需该步骤）。
   ```
   myVisualMapJ.add(radTool); // may not be required...
   ```
3. 将工具和 / 或工具栏添加到 GUI。通常，将工具添加到 MapToolBar，将 MapToolBar 添加到已经添加 VisualMapJ 对象的相同组件 （如同一 JPanel）。
   ```
   mapToolBar1.add(radTool);
   ```

语句的顺序也很重要。如果要地图工具与地图自动关联，则需要创建地图工具，将其添加到 MapToolBar，然后以该顺序将 MapToolBar 添加到和 VisualMapJ 相同的容器中。

此外，还可以将地图工具添加到菜单 （如通过调用 JMenu 对象的添加方法）。有关菜单上的地图工具显示的示例，请参阅在其地图菜单上显示工具的 MapXtreme Java 管理器。如果向工具栏和菜单添加了工具，则相应工具栏和菜单将保持同步；当用户选择工具栏上的工具时，相同的工具在菜单上也将显示为选中。

删除单独工具

删除单独工具可在 MapToolBar 中完成，方法是调用其 **removeMapTool** 方法。例如，如果要使用默认的 MapToolBar 对象，但是不想包含标尺工具，则可删除标尺工具，如下所示：

```
for (Enumeration e = mapToolBar1.getMapTools(); e.hasMoreElements(); )
{
   MapTool mt = (MapTool) e.nextElement();
    if (mt instanceof RulerMapTool) {
  mapToolBar1.removeMapTool(mt);
  break;
 }
}
```

此外，还可以构建只包含所要使用的地图工具的列表，然后将该列表传递到 MapToolBar 构造器。

配置和控制标准地图工具

在已将地图工具包括在应用程序之中后，可能需要调整工具的行为。配置选项的部分示例如下：

- 禁用地图工具。
- 重新配置 InfoMapTool，不显示其辅助 “信息工具”窗口。
- 控制是否将 RadiusSelectionMapTool 在注释图层 （Annotation 图层）中另存为圆。

一般而言，可通过设置地图工具的属性来控制其行为。可以设置的地图工具属性如下所示。如果使用可视化的开发环境支持属性页，则可通过使用适当的属性页编辑器来设置工具属性。否则，可以通过调用属性的适当设置方法，以编程方式来调整属性。例如，要配置启用的属性，可调用 setEnabled 方法。

常规设置

所有地图工具共享某些公共属性：

- **enabled** — 启用或禁用工具，调用其 **setEnabled** 方法。
- **cursor** — 指定定制光标，调用 **setCursor** 方法。
- **selected** — 选择工具，调用其 **setSelected** 方法。运行应用程序时，该工具将自动显示为选定。但是，这不会妨碍用户选择其他工具。

  ```
  radTool.setSelected(true);
  ```

放大 / 缩小设置

三个缩放工具（ZoomInMapTool、 ZoomInMapTool2 和 ZoomOutMapTool）可以通过以下属性自定义：

- **zoomFactor** — 此属性控制用户单击地图时地图的距离远近。默认设置为 2，换言之，如果地图显示 500 英里宽的范围，然后单击缩小，生成的地图将显示 1000 英里宽的范围
- **zoomMode** — 控制地图是否在用户点击的点重定中心，以及地图是否只在于相同位置保持对中时进行缩放。

RulerMapTool 设置

标尺工具提供以下属性：

- **calculationMethod** — 控制是使用笛卡尔坐标还是以球面坐标计算距离。
- **distanceUnit** — 指定用于显示距离的单位，例如公里或英里。
- **distanceWindowVisible** — 控制标尺工具是否打开辅助窗口以显示测量距离。要禁用辅助窗口，可调用 setDistanceWindowVisible(False)。

RulerMapTool 还具有一个限定 “距离”属性，用于在距离变更时通知监听程序。

InfoMapTool 设置

信息工具提供以下属性：

- **searchMode** — 控制 InfoMapTool 搜索哪些图层。例如，要搜索所有地图图层，可调用：

  ```
  myInfoMapTool.setSearchMode(SearchMode.FULL);
  ```

- **infoWindowVisible** — 控制是否打开辅助 “信息工具”窗口以显示由信息工具返回的文本。此时，可能需要禁用辅助窗口，在 GUI 中的其他地方显示生成的文本。要避免信息工具在辅助窗口中显示，可调用：

  ```
  setInfoWindowVisible(false)
  ```

在禁用信息工具窗口之后，需要在其他位置显示结果。创建一个实施 InfoObtainedListener 接口的对象 （表示该对象提供 infoObtained 方法），然后将监听程序注册到 InfoMapTool。在用户选择 InfoMapTool 并单击地图之后，将调用 infoObtained 方法，并传递 InfoObtainedEvent 对象。此外还可在 InfoObtainedEvent 对象上使用该方法，提取由 InfoMapTool 所返回的信息。此后，是否在 GUI 中的其他地方显示信息即取决于您。

SelectionMapTool 设置

MapXtreme 提供了若干种选择工具：BoundarySelectionMapTool、PolygonSelectionMapTool、 RadiusSelectionMapTool、 RectangleSelectionMapTool 和 ObjectSelectionMapTool。如果在应用程序中包括上述一个或多个工具，则用户即可选择选择工具，然后单击 （单击或拖动取决于工具）地图，选择地图上的一个或多个图元。

选择地图工具具有若干个可以配置的属性，如下所示。（并非所有属性均适用于 ObjectSelectionMapTool）。

- **saveAnnotation** — 布尔值，用于允许或禁止自动创建注释图层 （Annotation 图层），该图层的单一图元说明了所执行的选择区域边界。除 ObjectSelectionMapTool 之外的所有选择工具均允许保存注释。例如，RadiusSelectionMapTool 可以保存圆注释，表示用户选择的搜索半径。
- **searchType** — 控制用于确定是否应该选择一个图元的搜索标准。例如，如果一个区域只是部分位于搜索半径之内，则可能需要选择该区域，也可能不许选择该区域。要控制这一选项，可调用选择工具的 setSearchType 方法，然后指定以下值之一：

  SearchType.mbr — 返回其最小边界矩形和搜索区域交叉的图元。这是限定性最弱的搜索类型限制条件，返回的图元数量最多。

  SearchType.partial — 返回至少部分和搜索区域交叉的图元。

  SearchType.entire — 返回完全包含在搜索区域之内的图元 （默认），这是限定性最强的搜索类型。

  部分和全部都是绝对的，有效的搜索比较图元的真实几何范围和搜索区域。在需要真实和精确的结果时，必须使用这些搜索类型。 mbr 搜索是一个简化几何对象的近似方法，可实现更加迅速的比较。在特定数据源中，例如 Oracle Spatial。 mbr 搜索 (SDO_FILTER) 实际上是通过将图元的最小边界矩形和区域的最小边界矩形交叉来实施的，这样可以产生的 "命中数" 比使用实际搜索区域更高。
- **selectionMode** — 此属性控制搜索多少个图层。要控制搜索哪些图层，可以调用 setSelectionMode 并指定以下属性之一：

  SelectionMode.VISIBLE — 搜索当前可见的所有可选图层。此选项将跳过由于缩放而在当前不可见的图层。

  SelectionMode.FULL — 搜索所有可选图层。

SelectionMode.FIRST － 搜索所有可见和可选图层，但是当某一图层具有该搜索区域中的图元时，将会停止搜索。

- **selectionThemeType** － 控制是否创建 SelectionTheme 来突出显示所选图元，同时还控制专题如何给所选图元指定颜色。要控制所选图元的影线表示，可使用以下选项之一调用 setSelectionThemeType：

  SelectionMapTool.SELECTIONTHEME_OVERRIDE － 所选图元以影线表示，使用由 themeOverrideColor 属性的显式指定颜色。

  SelectionMapTool.SELECTIONTHEME_NONE － 不创建专题，所选图元的外观在选定之后不更改。

  SelectionMapTool.SELECTIONTHEME_DEFAULT － 所选图元将使用和原始图元颜色相反的颜色作影线表示。

- **themeOverrideColor** － 设置覆盖颜色 （用于在 selectionThemeType 设置为 SELECTIONTHEME_OVERRIDE 显式所选图元的颜色值）。例如，如果希望所有所选图元均显示为绿色，可调用 setThemeOverrideColor(Color.green)，然后调用 setSelectionThemeType (SelectionMapTool.SELECTIONTHEME_OVERRIDE)。
- **coordinateType** － 半径和矩形选择地图工具可用于指定选择区域 （用户绘制的圆或矩形）是使用屏幕坐标还是使用地图坐标。将此属性设置为以下值之一：

  ConfiningSelectionMapTool.COORD_MAP － 半径和矩形搜索工具将以地图坐标（即在屏幕上并非可见的圆的半径，该半径取决于地图投影，正如在某些地图投影中，地球本身并不显示为圆形一样，但是相应半径精确表示了距离中点指定距离中的全部区域）创建图元。

  ConfiningSelectionMapTool.COORD_SCREEN － 半径和矩形搜索工具将以屏幕坐标（如屏幕上指定半径的圆）创建图元。

在选择之后，所有 SelectionMapTools 均触发 SelectionToolEvents，以便在您创建和注册 SelectionToolListener 时，可以进行自定义。

创建定制 MapTools

如果所有标准 MapXtreme 地图工具均未提供所需的功能，您可以自行创建定制工具。要定义定制工具，您必须实施自己的 Java 类来满足 MapTool 接口的要求。在创建了此类之后，可以将定制工具实例化，然后将其添加到应用程序，其添加方式和添加标准工具的方式相同 （例如如上所示的半径工具）。

定制工具示例：SimpleRulerMapTool

在 /examples/client/beans 目录中提供了如何创建定制工具的示例，其中包含以下模块。

- **SimpleRulerMapTool.java** — 定制地图工具的示例。此类实施自定义的标尺工具，用于测量地图上的距离。和标准的标尺工具不同，这一定制工具在您停止移动鼠标时，将当前距离显示为 ToolTip。
- **SimpleRulerMapToolBeanInfo.java** — 一个 BeanInfo 类，控制如何展示工具属性（如在使用可视开发环境检查属性的时候）。
- **SimpleToolFrame.java** — 一个扩展 JFrame 的类。这是显示地图和工具栏的类。同时还是初始化 SimpleRulerMapTool 对象的地方。
- **SimpleToolApp.java** — 一个非常简单的类，提供主要的方法。

在开始研究定义 SimpleRulerMapTool 的代码之前，需要熟悉这些工具的具体操作。在此通过编译所有 4 个 Java 类，然后再运行 SimpleToolApp 类，即可看到定制工具的操作行为。

注：　在编译之前，可能需要编辑 SimpleToolFrame.java 来更改在 loadGeoset 调用中指定的文件名和 / 或路径，以便相应信息可以确定系统上的文件的名称和位置。

在该应用程序运行时，定制 SimpleRulerMapTool 将出现在工具栏末端。用户可以选择该工具，单击地图一次以开始测量距离、移动鼠标，然后暂停；用户暂停时，当前距离将显示为 ToolTip。用户可以再次单击，从其他位置开始测量。通过按下 Esc 键，用户可以取消当前线段。

MapTool 的基本要求

地图工具是操作对象，因此，可以将其添加到 JMenu 或 JToolBar 对象。由于地图对象是操作，因此对于您所创建的任意地图工具，都必须定义 **actionPerformed** 方法。在用户选择定制工具之后，无论是从工具栏还是从菜单，均将调用定制工具类的 actionPerformed 方法。

此外，您所创建的任意地图工具类都必须实施 **MapTool** 接口 (com.mapinfo.beans.tools.MapTool)。这意味着必须提供以下方法：

- **getCursor** — 返回和此 MapTool 关联的 java.awt.Cursor 对象。
- **isSelected** — 返回 True 或 False，表示当前是否选定此 MapTool。
- **setCursor** — 设置工具的光标对象。
- **setSelected** — 设置当前是否选定该工具。

　注：　选择一个工具将自动取消此前所选的地图工具。

有关这些方法的示例，请参阅 SimpleRulerMapTool.java、

在已经构建满足这些基本要求的定制工具之后，可以考虑向定制工具添加更多的功能。可以添加的其他功能在后续内容中作出了说明。

对于点击和拖动操作的响应。

用户使用地图工具可以和地图交互，为此大部分地图工具实施了 **MapMouseListener** 接口 (com.mapinfo.beans.vmapj.MapMouseListener)。实施此接口意味着提供以下响应各种不同鼠标操作的方法：

- **mouseClicked** － 在用户完成单击操作时调用 （按下然后释放鼠标按钮）
- **mouseDragged** － 在用户按住鼠标按钮并移动鼠标时调用
- **mouseEntered** － 在鼠标进入 VisualMapJ 显示区域时调用
- **mouseExited** － 在鼠标退出 VisualMapJ 显示区域时调用
- **mouseMoved** － 在没有按住鼠标按钮移动鼠标时调用
- **mousePressed** － 在用户按下鼠标按钮时调用
- **mouseReleased** － 在用户释放鼠标按钮时调用

SimpleRulerMapTool 类展示如何使用这些鼠标事件方法。此工具允许用户单击以设置测试距离的起点，因此 **mouseClicked** 方法包含存储起始坐标的代码。

接下来，用户可以移动鼠标，导致 SimpleMapRulerTool 显示弹性线。要生成这一弹性线效果， **mouseDragged** 和 **mouseMoved** 方法均调用 **handleStretch** 方法。

在 SimpleRulerMapTool 示例中，所包含的其他鼠标方法只用于满足 MapMouseListener 接口，因此相应方法为空。例如：

```
public void mouseEntered(MapMouseEvent e){}
```

取决于需要定制工具具备的具体行为，可向上述空方法添加代码。

键盘输入响应

在某些情况下，可能需要定制地图工具来处理键盘输入。例如 SimpleMapRulerTool 可用于在用户按下 Esc 键时取消弹性线的显示。

如果定制地图工具需要对键击作出响应，可实施 **KeyListener** 接口 (java.awt.event.KeyListener)，即实施以下方法：

- **keyPressed** － 在用户按下按键之后调用
- **keyReleased** － 在用户释放按键之后调用
- **keyTyped** － 在用户完成键的输出时调用 （按下和释放一个键）

SimpleMapRulerTool 检查 keyReleased 方法中是否有 Esc 键。如果已经按下改键，则 keyReleased 方法调用 cleanup 方法，令工具复位。

在 SimpleRulerMapTool 示例中，所包含的 keyPressed 和 keyTyped 方法只用于满足 KeyListener 接口，因此相应方法为空。

显示 ToolTip 文本

地图工具可以采用两种不同类型的 ToolTip 文本：

- 当用户将鼠标置于工具栏的工具之上时，将出现 ToolTip 显示工具的名称。要设置显示在此类 ToolTip 中的文本，可调用 putValue，例如：

  ```
  putValue(SHORT_DESCRIPTION,"Ruler");
  ```

- 当用户选择定制工具之后，光标位于地图之上，即可显示 ToolTip 文本。这一点可以通过提供 getToolTipText 方法，实施 **ToolTipTextSetter** 接口 (com.mapinfo.beans.tools.ToolTipTextSetter) 来完成。

例如，SimpleMapRulerTool 使用 ToolTip 文本显示当前光标位置和用户上次点击位置之间的距离。与此类似， SimpleMapRulerTool.java 提供了 getToolTipText 方法，该方法只返回表示距离测量结果的字符串 （如 123.45 米）。

注：即使定制工具实施 ToolTipTextSetter 接口，不调用 VisualMapJ 对象的 setShowToolTips 方法，也将无法看到 ToolTips。有关示例，请参阅作出以下调用的 SimpleToolFrame.java：

```
visualMapJ1.setShowToolTips(true);
```

在地图顶部绘制

某些地图工具在用户使用相应工具时对地图的外观作出了临时更改。要以此方式在地图上进行绘制，可实施 MapPainter 接口 (com.mapinfo.beans.vmapj.MapPainter) 及其 **paintOnMap** 方法。

例如， SimpleMapRulerTool 在地图顶部绘制一个弹性线，并当用户在其 paintOnMap 方法中移动鼠标时更新此线。

8

编写定制 Servlet

本章介绍如何使用 servlet （服务器端 Java 程序）部署地图绘制应用程序。

本章内容：

Servlet 概览

Servlet 是用于扩展 Web 服务器功能的 Java 组件。 Servlet 面向服务器， applet 面向浏览器，但是 servlet 没有图形用户界面。

MapXtreme Java 中的地图绘制引擎即部署为 servlet。它符合 Java 2 企业版体系结构，并且必须自 J2EE 验证的容器之内运行。借助于此， servlet 容器将管理负载平衡、容错等非地图绘制任务，以便 MapXtremeServlet 发挥最大效能 － 处理地图请求。

使用 Servlet

servlet 可以包含多种地图绘制功能，用于满足具体应用的需求。例如，可以构建提供基础地图导航的 servlet，提供平移、缩放和测量两点距离等功能。如果需要更加复杂的应用程序，可以考虑提供选择功能或专题地图绘制功能。

注： 在编译 servlet 源程序之前，需要配置 Java IDE，以便在 IDE 的 classpath 中包括 J2EE servlet 类文件 （如 servlets.jar）。

MapXtreme Java 适用于两至三层的 Web 应用程序。两者之间的区别在于 MapJ 组件的位置是在服务器还是在客户机。后续内容介绍了各种不同的配置选项。servlet 体系结构的示例可参阅位于以下网址的 MapInfo 知识库：**http://testdrive.mapinfo.com/kbase_by_product**。

瘦客户机 / 胖服务器

以下说明介绍了最常见的配置 － 浏览器作为客户机，用户定义的 servler 使用中间层的 MapJ 对象， MapXtremeServlet 位于中间层，数据库位于最后一层。应用程序可利用 servlet、 Java 服务器页或 Enterprise JavaBean 的任意组合。

当客户机通过浏览器发出请求之后， Web 服务器将该请求转发至用户定义的 servlet，然后即可更新 MapJ 对象的状态。 MapJ 对象随后将用于与 MapXtremeServlet 沟通地图绘制请求。如果是图像请求， MapXtremeServlet 将地图的栅格图像返回给用户定义的 servlet。用户定义的 servlet 随后即可将此图像嵌入到 HTML 页面，并将该页面返回给最终用户的浏览器。

在中间层，用户定义的 servlet 还可以利用 MapXtreme 的 servlet 库来帮助您构建 HTML 页面。例如，现有可以使用基于图层控制的方式，利用库方法来创建 HTML。

这个三层的体系结构具有以下特征：

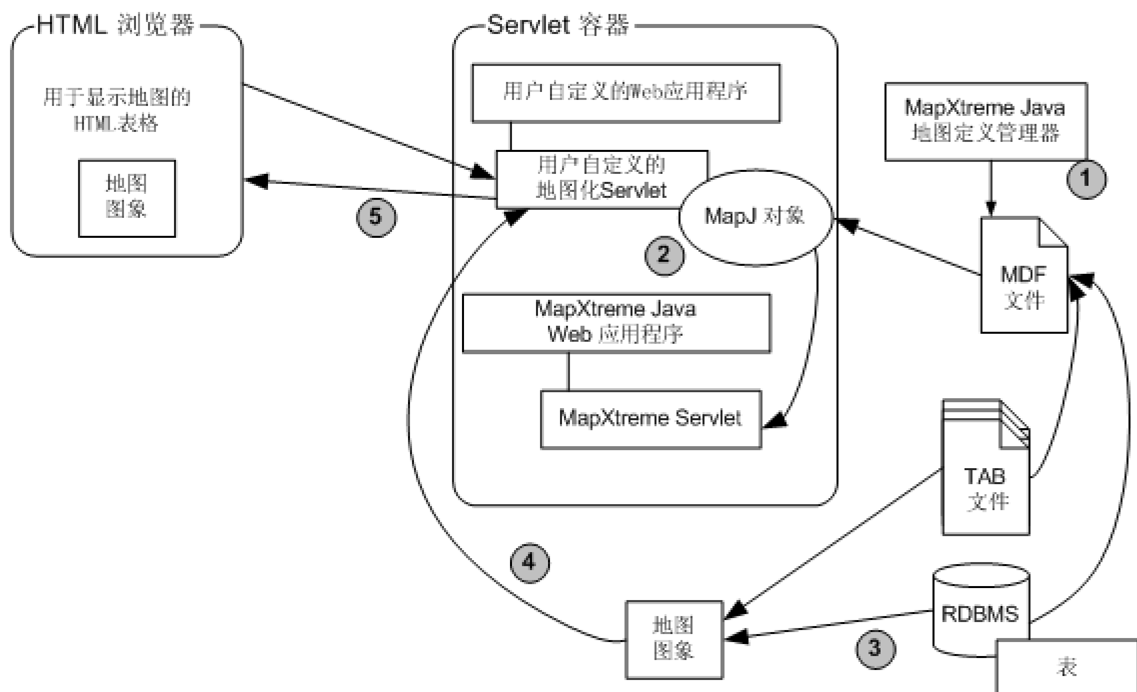
- MapJ 部署在用户定义的 servlet 的中间层中。
- MapXtremeServlet 部署在中间层。
- 客户端不需要 Java。客户机可以发送 HTTP 请求，并接收作为响应的 HTML 页面。
- 生成的网络流量最低：无需 Applet，即不用下载 applet。向量数据不发送到客户机，发送到客户机的只是嵌入栅格图像的 HTML 页面。类似 GIF 的栅格格式通常生成的地图图像大小为 15ñ25K。

这些特征令三层体系结构成为适用于因特网部署的理想方案，此时 Web 应用程序开发人员对客户机配置具有的控制最少。这一部署是 “公共特征最少” 的途径，可用于来满足那些没有支持 Java 的浏览器和 / 或网络带宽太小的客户机的需求。

此外， MapXtreme Java 提供了定制 JSP 标记库，可用于在快速应用程序开发 (RAD) 环境下创建中间层 servlet，例如在 MapXtreme Java 管理器中的 Web 应用程序构建器环境下。

瘦客户机体系结构

本节介绍瘦客户机的体系结构。



下表列出了这一体系结构的要点。

客户端	通信	服务器端
HTML 浏览器	HTTP 协议	**MapXtreme Java servlet** t 使用服务器资源远程渲染 用户定义的 **Servlet** t 接受并发客户机连接 t 每个连接客户机均分配有专用的 MapJ 对象 tMapJ 保存到会话 t 本地 MDF 文件带有 LocalDataProviders

以下步骤和瘦客户机体系结构图中的编号相对应：

1. 在使用本地数据提供方访问本地数据源的服务器计算机上创建 MDF 文件。
2. 为每个连接客户机创建的 MapJ 对象将使用 MDF 文件初始化。
3. 用户定义的 servlet 使用 MapXtremeServlet 来构建地图图像。
4. 地图图像返回到用户定义的 servlet。
5. 用户定义的 servlet 地图将图像集成到 HTML 中，用于返回到客户机。

有关瘦客户机体系结构的要点小结如下：

- 无需为客户机编程。只是返回由浏览器解释。
- 通信通过 HTTP POST 请求完成。唯一的例外是第一个请求，该请求是 HTTP GET 请求。
- MapJ 驻留在服务器上用户定义的 servlet 之中。
- 远程渲染在服务器端完成。
- 服务器上的 MDF 文件使用 LocalDataProviders。

胖客户机 / 瘦服务器

在两层或胖客户机配置中， MapJ 对象和业务逻辑均在客户机端部署，通常是作为浏览器中的 applet。此类部署的主要优点是可使用 MapXtreme Java 版的 JavaBean。与使用低级 MapJ API 相比，在可视化的 RAD 环境中，使用 MapXtreme Java 版 JavaBean 可以更加迅速地创建应用程序。 MapXtreme Java 版的 JavaBean 提供了可视化的地图工具、工具栏、向导，可直接纳入到应用程序之中使用。

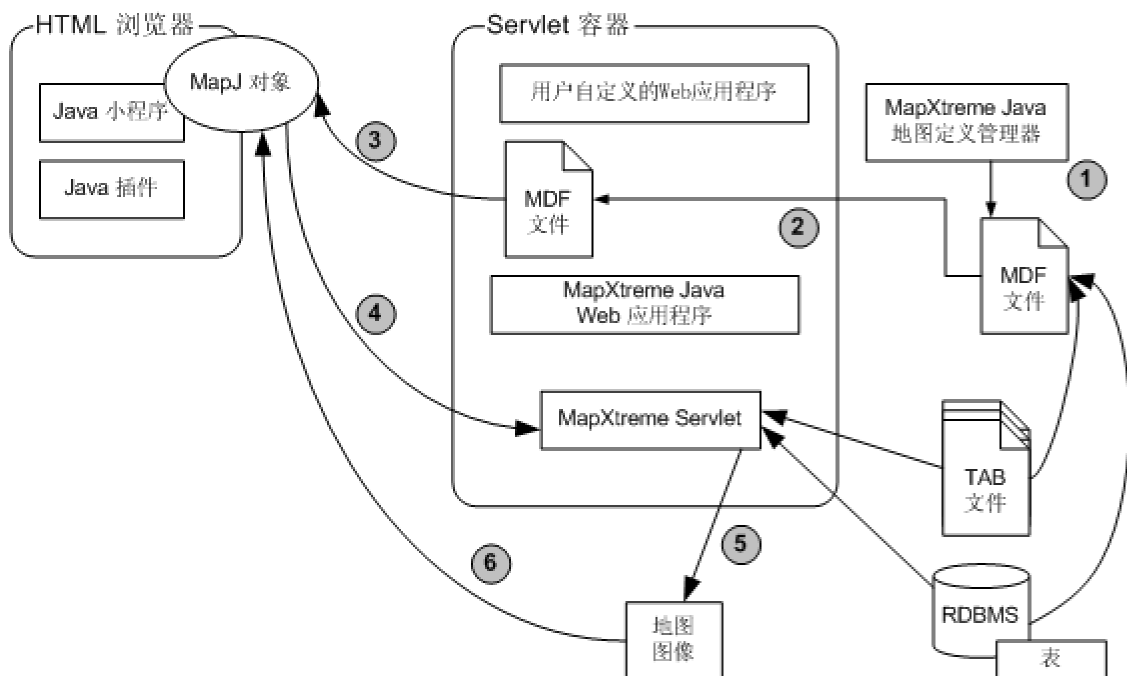
在两层部署中，客户机将从 Web 服务器先下载一个包含 JavaBean 的 applet。该两层体系结构具有以下特征：

- MapJ 部署在客户端的 applet 中。
- MapXtremeServlet 将部署在中间层，用于地图渲染和数据获取。
- 客户机上需要 Java：客户机浏览器必须支持 Java 2 平台 VM 1.4.0 或更高版本（或具有适当的插件）。
- 网络流量较大：必须下载包含 JavaBean 的 applet。向量数据可发送到客户机；此外，与栅格文件的大小相比，向量数据的大小变化较大。

由于存在这些特征，两层体系结构最适合用于内网部署，此时的部署环境更加趋同并且易于控制。当渲染和数据访问由 applet 完成时，需要较高的网络带宽。客户机也需要采用性能较高的计算机。

胖客户机体系结构

本节介绍胖客户机的体系结构。



下表列出了这一体系结构的要点。

客户端	通信	服务器端
tHTML 浏览器 tapplet 采用 VMapJ tMapJ 在客户端	tHTTP 协议 t 用于远程渲染 t 用于远程数据访问	**MapXtreme Java servlet** t 使用服务器资源远程渲染 t 通过 MapXtremeDataProvider 获取数据 t 采用下载到客户端的 MDF 文件

以下步骤和胖客户机体系结构图中的编号相对应：

1. 在使用 MapXtremeDataProvider 访问本地数据源的服务器计算机上创建 MDF。
2. 将 MDF 文件置于 applet 所在的 Web 服务器的公共访问目录中。
3. 从 Web 服务器将 MDF 内容传递到 applet。

   注：　MDF 必须使用每个图层的 MapXtremeDataProviderRefs。
4. 将 MapJ 设置发送到服务器，用于数据源访问和地图图像创建。
5. 使用服务器上的数据源来创建地图图像。
6. 将图像传送回客户机用于显示。

有关胖客户机体系结构的要点小结如下：

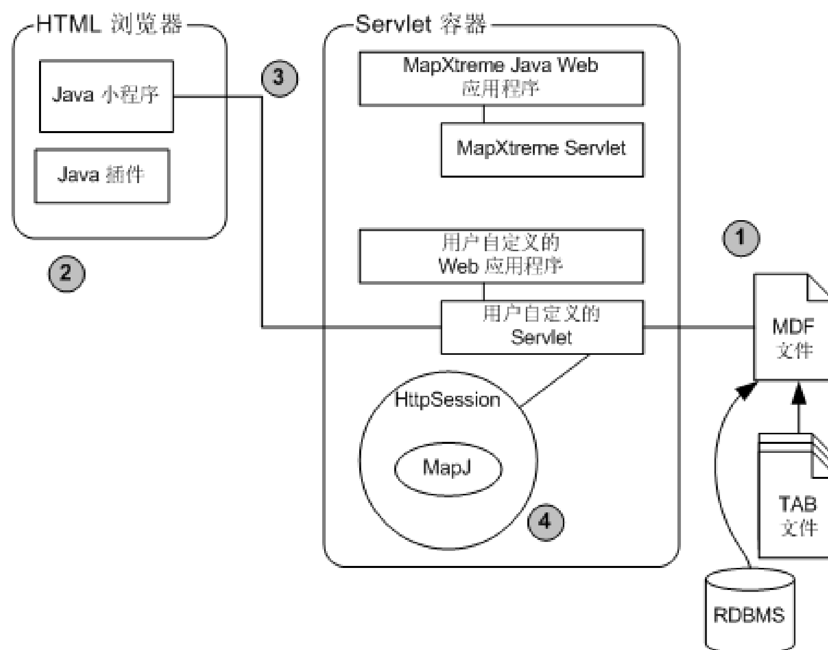
- 客户机包含地图绘制功能并需要大量编程。
- MapJ 驻留在客户端。
- MDF 文件下载到客户端 applet，以便只使用 MapXtremeDataProviders 定义其图层。
- 远程渲染在服务器上完成，生成的图像发送回客户机。

中型客户机 / 中型服务器

在中型客户机 / 中型服务器配置中，可配置 applet 遍历用户定义的 servlet 以获取其数据和地图图像。这一配置专用于 applet 充当客户机和图层数据源数据包含在 RDBMS 中的情况。在上述情况下，中间层中由用户定义的 servlet 将提供一个间接层面，以便允许 RDBMS 访问。

中型客户机体系结构

本节介绍中型客户机的体系结构。



下表列出了这一体系结构的要点。

客户端	通信	服务器端
tHTML 浏览器 tapplet	tHTTP 协议 t 获取 sessionID 的握手请求 t 采用串行化对象传递数据	**MapXtreme Java servlet** t 使用服务器资源远程渲染 用户定义的 **Servlet** t 接受并发客户机连接 t 每个连接客户机均分配其自己的 MapJ 对象 tMapJ 保存到会话 t 采用带有 LocalDataProviders 的 MDF 文件

以下步骤和中型客户机体系结构图中的编号相对应：

1. 在服务器上创建的 MDF 文件为每个图层使用本地数据提供方。
2. HTML 浏览器下载 applet HTML 页以开始执行 applet。
3. 串行化 Java 对象用于在 applet 和用户定义的 servlet 之间发送信息。

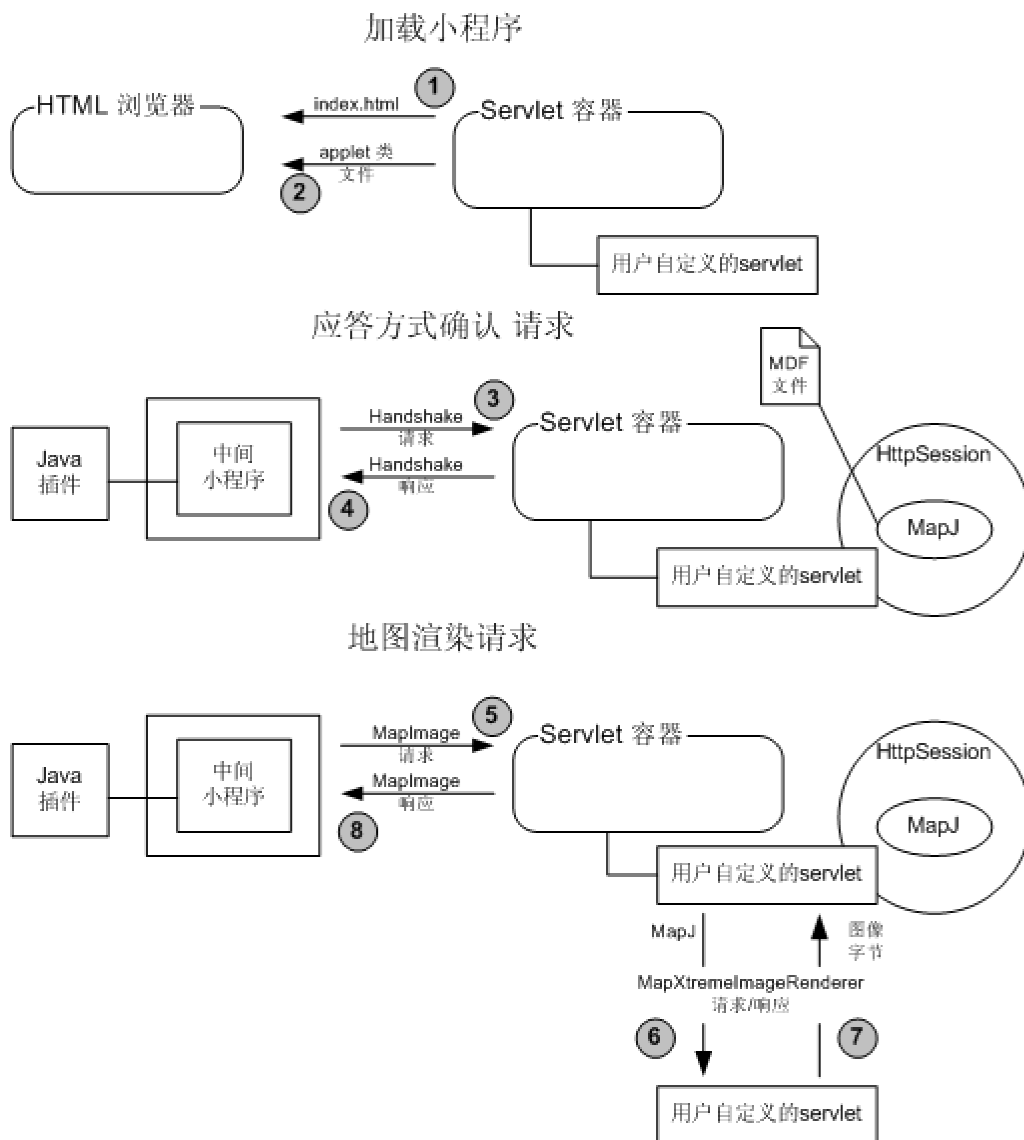
4. 将为每个连接的客户机使用用户定义的 servlet 创建 HTTPSession，用于保存客户机的 MapJ 对象。

有关中型客户机体系结构的要点小结如下：

- 客户机需要编程，但是仍然绘制 / 修改通过编程方式方式保存在服务器上的地图。
- 通信通过串行化对象完成。在客户机 / 服务器之间传递更多具有实际意义的数据。
- MapJ 驻留在服务器端。
- 远程渲染在服务器端完成。
- 服务器上的 MDF 文件使用 LocalDataProviders。

中型客户机通信

本节介绍在中型客户机 / 中型 servlet 体系结构下，在客户机 applet 和用户定义的 servlet 之间的通信过程。



以下步骤和中型客户机通信示意图中的编号相对应：步骤 1-4 只执行一次，步骤 5-8 根据需要可重复多次。

1. 当 index.html 文件下载到浏览器之后，将强制浏览器 （通过特殊 HTML 标记）下载 applet 的类文件并开始执行相应文件。
2. 所要下载的文件仅限于 Java 2 平台标准版 SDK 1.4.0. 版中的文件。在客户端 applet 中不使用与 MapInfo 相关的专有类文件。
3. 握手请求是从客户机发送到用户定义的 servlet 的第一个请求。该请求告知 servlet 为有关客户机创建新的 servlet 会话，返回特殊的 URL，以便客户机在进行后续请求时使用。这一特殊的 URL 包含嵌入的会话 ID 值，服务器将使用该值来获取客户机的特定 servlet 会话。 servlet 会话只是存储在内存 （或者如果会话是永久性的，则为文件系统上的文件）中的数据段，其中包含客户机的状态信息。
4. 握手响应将从用户定义的 servlet 发送回客户机。其中包含具有嵌入会话 ID 值的特殊 URL，以便客户机在对服务器发出后续请求时，使用这一新的 URL。
5. 在从客户机发送的一个请求 － 握手请求之后，每个发送的请求都将是 MapImage 请求。MapImage 请求包含图像的说明，该图像将由 MediumServer 创建和返回。此外，该请求中还提供了更多详细信息，例如要创建的图像的像素尺寸，以及需要预先应用的任意图像导航操作 （缩放、平移等）。
6. 在根据需要修改 MapJ 对象之后，用户定义的 servlet 将向 MapXtremeServlet 请求地图渲染。这称为远程渲染。
7. 在 MapXtremeServlet 已经生成地图图形之后，它会将图像数据发送回用户定义的 servlet 以便返回给客户机。
8. 用户定义的 servlet 将 MapImage 响应发送回客户机，其中包含地图图像本身的数据。客户机 applet 随后将从响应消息中提取图像数据，并在 GUI 上绘制地图图像。

MapXtremeServlet 数据流

在开发 MapXtreme Java 4.x 支持的客户机应用程序时，用户地图的状态将保存在类类型 com.mapinfo.mapj.MapJ 的对象中。在运行时，可对该对象执行渲染操作和搜索操作，以便在应用程序中实现真正的地图绘制功能。基于 MapJ 对象中的设置以及 MapJ 对象的使用方式的不同，介于客户机应用程序和 MapXtremeServlet 之间的数据流情况将有所差异。

导致数据流情况有所差异的一般因素如下所示：

- 在 MapJ 对象的图层中使用的是什么类型的数据提供方？
- 是否正在渲染地图图像？
- 是否正在对图层执行搜索？
- 如果渲染地图图像，那么是采用本地渲染，还是采用远程渲染？

数据提供方操作

本节介绍与数据提供方类型和执行操作有关的各种数据流情况。

LocalDataProviderRef：远程渲染

使用 LocalDataProviderRef 进行远程渲染的步骤如下。

1. 客户机将其 MapJ 对象发送到 MapXtremeServlet。
2. MapXtremeServlet 直接读取数据源数据。
3. MapXtremeServlet 使用此数据创建地图图像。
4. MapXtremeServlet 将地图图像数据返回客户机。

LocalDataProviderRef：本地渲染

使用 LocalDataProviderRef 进行本地渲染的步骤如下。

1. 在客户机之内， MapXtreme Java 类直接读取数据源数据。
2. 在客户机之内， MapXtreme Java 类使用此数据创建地图图像。
3. 在客户机之内，地图图像随后将转录到图形上下文。

LocalDataProviderRef：图层搜索

使用 LocalDataProviderRef 进行图层搜索的步骤如下。

1. 在客户机之内， MapXtreme Java 类直接读取数据源数据。
2. 在客户机之内， MapXtreme Java 类使用此数据执行图层搜索。
3. 在客户机之内， MapXtreme Java 类生成搜索结果。

MapXtremeDataProviderRef：远程渲染

使用 MapXtremeDataProviderRef 进行远程渲染的步骤如下。

1. 客户机将其 MapJ 对象发送到 MapXtremeServlet。
2. MapXtremeServlet 通过 MapXtremeServlet 间接读取数据源数据，此时 MapXtremeServlet 是通过 MapXtremeDataProviderRef 的 URL 定位的。

   这些 MapXtremeServlet 实例可能会有所不同，并驻留在不同的计算机上。如果 MapXtremeDataProviderRef URL 只引用当前的 MapXtremeServlet 实例 （如引用 MapJ 对象所发送的实例），那么直接访问数据源数据的将是这一当前的 MapXtremeServlet 实例。
3. MapXtremeServlet 使用此数据创建地图图像。
4. MapXtremeServlet 将地图图像数据返回客户机。

MapXtremeDataProviderRef：本地渲染

使用 MapXtremeDataProviderRef 进行本地渲染的步骤如下。

1. 在客户机之内， MapXtreme Java 类通过 MapXtremeServlet 间接读取数据源数据，此时 MapXtremeServlet 是通过 MapXtremeDataProviderRef 的 URL 定位的。
2. 在客户机之内， MapXtreme Java 类使用此数据创建地图图像。
3. 在客户机之内，地图图像随后将转录到图形上下文。

MapXtremeDataProviderRef：图层搜索

使用 MapXtremeDataProviderRef 进行图层搜索的步骤如下。

1. 在客户机之内， MapXtreme Java 类通过 MapXtremeServlet 间接读取数据源数据，此时 MapXtremeServlet 是通过 MapXtremeDataProviderRef 的 URL 定位的。
2. 在客户机之内， MapXtreme Java 类使用此数据执行图层搜索。
3. 在客户机之内， MapXtreme Java 类生成搜索结果。

Servlet 转发

为了利用在 J2EE 2.2 应用程序中提供的 servlet 转发功能， MapXtreme Java 专门提供了 **IntraServletContainerRenderer**。此特性为将栅格图像返回到客户机提供了一种可选途径。这一渲染器在渲染器和 MapXtremeServlet 之间不需要套接字连接，但是这对于 MapXtremeImageRenderer 而言却是不可或缺的。

这一部署选项的优点在于栅格图像可以直接发送到客户机。 MapXtremeServlet 不需要将图像写入到中间层，然后再令中间层将其重写回客户机。但其限制是应用程序必须部署在和 MapXtremeServlet 相同的容器之中。在进行应用程序规划时，务必考虑这一因素。

IntraServletContainerRenderer 构造器将输入该信息视为必要，以便中间层 servlet 获取 MapXtremeServlet 的 **RequestDispatcher** 对象。 RequestDispatcher 对象处理 servlet 转发必需的信息如下所示：

- 由 com.mapinfo.mapxtreme.MapXtremeServlet 使用的别名，例如 mapxtreme47
- MapXtremeServlet 的 ServletContext 对象，或是 servlet 上下文的 URI，例如 /mapxtreme47
- MapXtremeServlet 将使用 HttpServletRequest 和 HttpServletResponse 对象满足该请求

- 栅格图像的 mime 类型
- 图像是否应该为多个部分，以及多个部分的更新间隔

有关 IntraServletContainerRenderer 的详细信息，请参阅*第 11 章：渲染涉及的考虑因素* 。

示例 Servlet：HTMLEmbeddedMapServlet

HTMLEmbeddedMapServlet 是 MapXtreme Java 附带的示例 servlet。该文件可见于 /examples/server/Java/servlet。此外， mxjserversamples.jar 中还提供了预编译的版本。

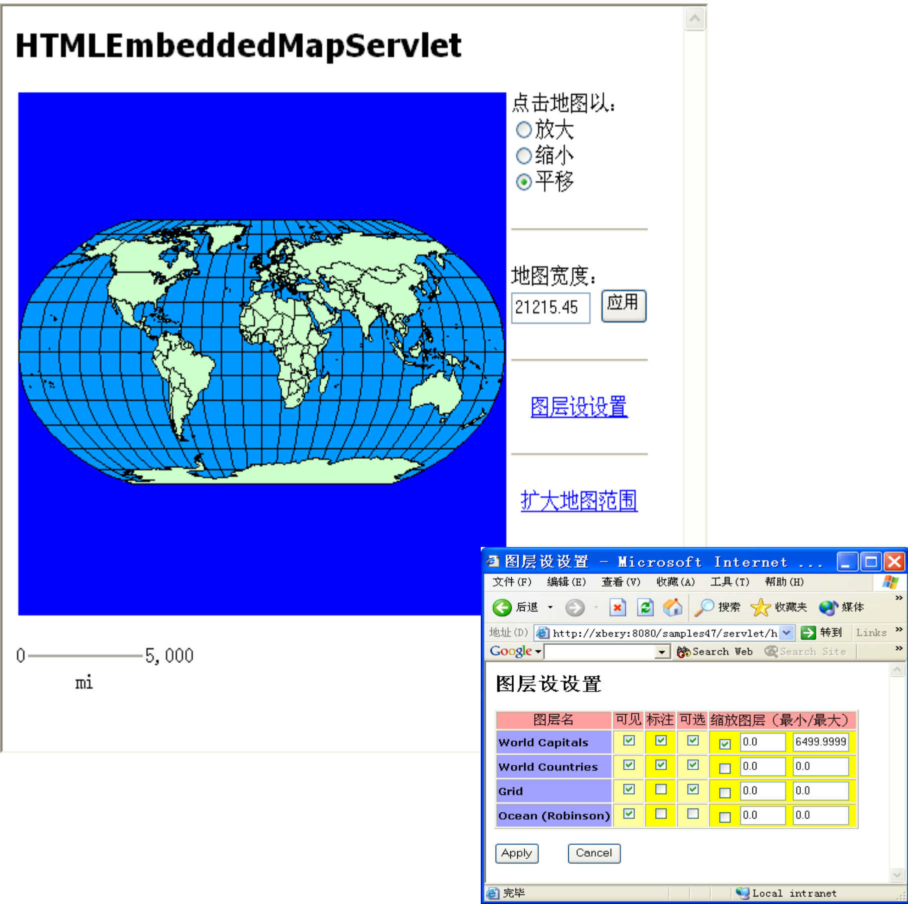
HTMLEmbeddedMapServlet 提供了嵌入这些基本地图绘制元素的 HTML 页面。

- 显示地图的地图框架
- 用于缩小、放大、平移的单选按钮
- 地图宽度框，用户用于键入新地图的宽度
- 应用按钮，用于应用地图宽度
- 图层设置链接，显示可以启用或禁用的图层的表
- 切换，用于放大或缩小地图
- 比例尺

通过打开浏览器，然后键入如下所示的 MapXtremeServlet URL，即可运行 HTMLEmbeddedMapServlet：

```
http://stockholm:8080/samples47/htmlmap
```

URL 可能会因具体设置方式而有所不同。



要自定义这一 servlet，可通过修改 HTMLEmbeddedMapServlet.java 中的适当变量并重新编译来更改地图的宽度或高度。在修改和重新编译 servlet 时，需要将其 类文件复制到适当目录。例如，对于 Tomcat，可能需要将 .class 文件复制到 Tomcat 的 webapps/samples47/WEB-INF/classes 目录 （如果是从 jar 运行，可能需要将类复制到 WEB-INF/lib/ mxjserversamples.jar）。

如果编译样例，务必要删除、移动或重命名 mxjserversamples.jar。否则，相应的 servlet 容器可能会使用源自 jar 文件的 HTMLEmbeddedMapServlet.class 文件，而不是使用您所编译的类文件。

HTMLEmbeddedMapServlet 代码示例可更改众多设置而无需重新编译。 servlet 使用标准的 servlet 初始化参数加载其众多设置。例如，要加载的地图名称 （如 world.gst）可使用初始化参数覆盖。这样，如果只需更改要加载的地图的名称，则只要编辑 init 参数即可，无需修改 servlet 源代码，。

下表介绍 HTMLEmbeddedMapServlet 示例预期将要用到的最重要的初始化参数。有关由示例 servlet 所用初始化参数的完全列表，请查看 HTMLEmbeddedMapServlet.java 中的注释。

初始化参数	说明	示例
filetoload	将要显示的地图文件 （.gst 或 .mdf 文件）的完全路径。	C:/mxt/maps/world.gst
mappath	geoset (.gst) 地图文件在服务器上安装目录的路径。不适用于地图定义。	C:/mxt/maps
mapxtremeurl	MapXtremeServlet URL	/mapxtreme47/mapxtreme

如果所用 servlet 容器提供管理员工具 （如 JRun 或 JavaWebServer），可使用该工具来设置初始化参数，如上所示。某些 servlet 容器可能需要在 XML 文件中指定初始化参数，后续内容对此作出了有关说明。

在 Tomcat 中编辑初始化参数

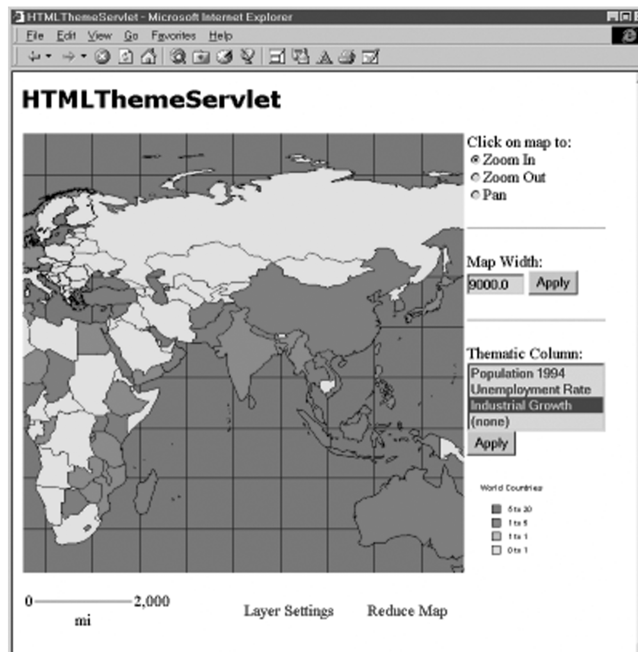
修改用于 Tomcat 的初始化参数可通过使用文本编辑器编辑 web.xml 文件来实现。以下示例显示了定义 HTMLEmbeddedMapServlet 的三个初始化参数的 <servlet></servlet> 块，这三个参数分别是 mappath、 filetoload 和 mapxtremeurl init。

```
<servlet>
  <servlet-name>
    htmlmap
  </servlet-name>
  <servlet-class>
    HTMLEmbeddedMapServlet
  </servlet-class>
  <init-param>
    <param-name>
      mappath
    </param-name>
    <param-value>
      C:\mxt\maps
    </param-value>
  </init-param>
  <init-param>
    <param-name>
      filetoload
    </param-name>
    <param-value>
      C:\mxt\maps\world.gst
    </param-value>
  </init-param>
```

```
    <init-param>
      <param-name>
        mapxtremeurl
      </param-name>
      <param-value>
        http://hostname/mapxtreme47/mapxtreme
      </param-value>
    </init-param>
  </servlet>
```

带有专题功能的样例 Servlet

在位于 /examples/server/java/thematic 的示例目录下还提供了增加版本的示例 servlet HTMLEmbeddedMapServlet，该示例集成了专题影线表示功能。有关向 servlet 添加专题支持的详细信息，请参阅 **HTMLThemeServlet.java** 文件。



使用 Servlet 实用程序库 (MapToolkit)

MapXtreme Java 为构建 servlet 提供了帮助程序方法库。借助于 MapJ 对象， MapToolkit 类可以帮助您构建支持地图的网页公共元素，例如图层控制表单。在示例应用程序 HTMLEmbeddedMapServlet 中使用的方法就取自此类。

注： 此库只适用于编写 servlet 源代码的用户。大部分开发人员喜欢使用 JSP 应用程序，如果计划在 JSP 中编写应用程序，就无需再使用 servlet 实用程序库。但如果出于某些原因无法使用 JSP 库 （如客户机无法符合 JSP 库所强制的 JavaScript 要求），那么就可能需要使用到 servlet 实用程序库。

使用此类来简化 servlet 的开发。该库中包括以下元素：

地图工具 — 单选按钮，确定地图导航元素 （缩、放、平移）。

缩放框 — 文本字段，显示当前的地图缩放，允许用户键入新的缩放。

图层控制 — HTML 页面，显示地图中的图层，允许用户选中或清除用于可选性、可见性和自动标注的设置。

比例尺 — 可查看的元素，显示地图的比例。

地图大小切换 — 用于放大或缩小地图大小的链接。

Servlet MapToolkit 中的方法

MapToolkit 提供了以下方法。有关 MapToolkit 类的完全说明，请参阅安装在计算机的 mapxtremejava/docs/devsupport 目录下的 HTML 参考资料。

方法	说明
getHTMLLayerListControl	返回表示用作图层控制对话框的 HTML 页的字符串。
applyLayerSettings	更新地图以反映用户在图层控制页中选择的所有选项。
getHTMLZoomControl	返回表示缩放控制的字符串 — 一组 HTML 标记，提供文本字段用于显示当前地图缩放宽度；以及 “提交” 按钮，应用用户所键入的新缩放宽度。
getHTMLMapToolsControl	返回字符串表示一组用于缩、放、平移的单选按钮。
getHTMLScaleBar	返回的字符串表示定义地图比例尺的 HTML 语法。指定特定宽度，或指定 0，此时比例尺将调整并舍入至大约地图图像宽度的 1/4。
getToolNumber	返回整数，表示相应于工具名称的工具编号。
getStr	从资源包返回字符串资源。

第 2 部分：MapJ API 简介

本开发人员指南的第 2 部分介绍如何使用 MapJ API 来创建和控制应用于应用程序的地图信息。

主题：

- 第 9 章：MapJ API
  本章介绍所有 MXTJ 地图绘制应用程序的起点：MapJ 对象。本章定义 MapJ，如何创建基本地图、渲染图像、控制地图视图，此外还介绍图元、样式和专题。
- 第 10 章：在图层中绘制地图
  本章介绍地图的基本元素 — 存储地图图元和信息的图层的集合。此外还介绍了特殊图层：Annotation 和 Raster 图层。
- 第 11 章：渲染涉及的考虑因素
  本章提供了有关 MapXtremeImageRenderer 和 LocalRenderer 以及其他渲染选项的更多信息。
- 第 12 章：访问远程数据
  本章介绍更加有效的远程数据访问方式 — 数据库连接池。
- 第 13 章：图元和搜索
  本章定义图元和 FeatureSets，同时提供了在寻找信息时可以采用的多种搜索方法。
- 第 14 章：标注和样式
  此外还提供了有关标注自定义和控制其可见性和位置的详细信息。并介绍了用于图元和标注的样式。
- 第 15 章：专题地图绘制和分析
  要获取有关地图的更多详细信息，可使用功能强大的专题地图绘制图元，以便对数据作出可视化的比较，进而作出适当的决策。

9

MapJ API

本章介绍 MapJ API 和要在后续章节中详细说明的主题。

本章内容：

MapJ 对象

MapJ 是一个便捷小巧的组件，提供了通过 MapXtremeServlet 或通过其本身来创建地图的界面。MapJ 可以发出两种类型的请求。一种请求称为图元的向量格式数据，一种称为请求地图图像文件。MapJ 的使用目的在于维护地图状态，其中包括保持图层、坐标系、距离单位和地图边界等记录。

MapJ 对象可以配置用于与不同类型的渲染器和 DataProvider 协同工作。在最为典型的配置中， MapJ 是 MapXtremeServlet 的客户机。 MapJ 向 MapXtremeServlet 实例发送请求，请求的一部分为 servlet 提供了其当前状态。 MapJ 从 servlet 获取地图图像和数据。

MapJ 还可以独立工作，直接获取地图数据并生成地图图像。MapXtreme 采用了基于组件的设计方案，其强大之处在于可配置为与其他变更版本协同工作。例如， MapJ 可配置为通过一个或多个 MapXtremeServlet 实例访问地图数据，同时仍然可以显示地图图像。有关配置选项的详细信息，请参阅*第 3 章：应用程序规划* 。

随产品提供的 MapXtreme Java 对象模型通告几乎显示了自 MapJ 对象派生的每个对象、属性和方法。显示在 MapJ 对象之下的每个方法均可用于帮助构建整个 MapJ 对象。每个 MapJ 对象主要由 Data Provider、 Layers 和 Feature 对象定义。MapXtreme 示意图上的其他对象均将有助于创建和渲染 MapJ 对象，例如 Data Provider 对象和 Renderer 对象。

创建第一个地图

后续内容将介绍使用 MapJ API 创建地图的一般过程。在此情况下， MapJ 通信时将其请求发送到 MapXtremeServlet。以下提供了相关的说明步骤以及其所在的页码。

1. 第 154 页的 *MapJ 对象初始化* 。
2. 第 155 页的*加载地图数据* 。
3. 第 156 页的*设置地图设施边界* 。
4. 第 157 页的*渲染地图* 。

MapJ 对象初始化

在应用程序中使用 MapJ 对象之前，必须先初始化 MapJ。要初始化 MapJ，只需采用以下 Java 代码即可：

```
myMap = new MapJ();
```

加载地图数据

在创建 MapJ 对象之后，还必须加载地图数据。

此时可加载 geoset 或地图定义。有关打开和创建地图定义和 geosets 的详细信息，请参阅 *第 5 章：管理 MapXtreme Java* 。

在 MapXtreme Java 版中，没有默认的地图定义。因此，作为初始化过程的一个组成部分，必须设置默认的地图定义。

加载 Geoset

以下是加载 geoset 文件的示例：

```
myMap.loadGeoset(geosetName, dataDir, servletURL);
```

其中

geosetName 是到 geoset 的完整路径

dataDir 是 geoset 中引用的 .tab 文件在服务器计算机上的位置 （可能和 MapXtremeServlet 不在相同的计算机）

servletURL 是 MapJ 使用远程 DataProviderRef 时到 MapXtremeServlet 的路径 （如果使用 LocalDataProviderRef，参数为空）。

例如：

```
myMap.loadGeoset("c:\\mapxtreme\\maps\\world.gst",
    "c:\\mapxtreme\\maps", "http://stockholm:8080/mapxtreme47/
    mapxtreme");
```

加载地图定义

此时要加载的即可以是存储在文件中的地图定义，也可以是存储在数据库中的记录或存储命名资源库中的命名地图。

要加载地图定义，必须先创建 MapDefContainer， MapDefContainer 是地图定义存储位置的抽象表示。

要为存储在文件中的地图定义创建 MapDefContainer，可使用以下代码：

```
MapDefContainer mdc = new FileMapDefContainer(dir)
```

其中 `dir` 是到包含地图定义文件目录的完整路径。

例如：

```
MapDefContainer mdc = new FileMapDefContainer("c:\\mapxtreme\maps")
```

要为存储为 RDBMS 记录的地图定义创建 MapDefContainer，可使用以下代码：

```
MapDefContainer mdc = new JDBCMapDefContainer(
    driver, url, user, password)
```

其中 driver°¢url°¢user 和 password 均为数据库连接参数。

以下是从 Oracle RDBMS 进行加载的示例：

```
MapDefContainer mdc = new OraSoMapDefContainer(
    "oracle.jdbc.driver.OracleDriver",
    "jdbc:oracle:thin:@machinename:1521:dbSid", "username",
    "password", "tableName", "Name", "Map_Definition");
```

要创建用于命名资源库中存储的地图定义的 MapDefContainer，可使用以下代码：

```
MapDefContainer mdc = new NamedMapDefContainer(providerUrl, path);
```

其中 providerUrl 为 file:/// 或指向命名资源库的根的 http:// URL， path 为到命名地图的相对于命名资源库的根的路径。

例如：

```
MapDefContainer mdc = new NamedMapDefContainer(
    "http://torpedo:8080/mapxtreme47/namedresource", "mymaps");
```

将通过以上 URL 所指的 NamedResourceServlet 引用 mymaps 中的命名地图。

要加载命名地图定义，可使用以下代码：

```
myMap.loadMapDefinition(mapDefContainer, name)
```

其中 **mapDefContainer** 是以上定义的容器类， **name** 是要从容器加载的地图 （在 saveMapDefinition 命令中使用的名称）。

例如：

```
myMap.loadMapDefinition(mdc, "Asia");
```

设置地图设施边界

使用 **MapJ.setDeviceBounds()** 设置渲染的地图图像的大小。这是在地图渲染之前设置的。设施边界设置渲染器返回图像的尺寸的像素数。例如，需要返回的是 800x600 的地图。默认的图像大小是 640x480。

要设置设施边界，可使用 MapJ 对象的 setDeviceBounds 方法。如果接受默认边界，则无需调用此方法。

```
myMap.setDeviceBounds(new DoubleRect(0, 0, 800, 600));
```

渲染地图

要渲染地图，必须将 renderer 对象实例化。以下示例使用 MapXtremeImageRenderer，并将图像渲染为 GIF 文件。

指定指向 MapXtreme servlet 的 URL，该 servlet 将远程连接到地图引擎。

```
String mapxtremeServletUrl =
    "http://stockholm:8080/mapxtreme47/mapxtreme";
```

创建 ImageRequestComposer：

```
ImageRequestComposer imageRC =  ImageRequestComposer.create(
    myMap, 256, Color.blue, "image/gif");
```

创建 MapXtremeImageRenderer：

```
MapXtremeImageRenderer renderer =
    new MapXtremeImageRenderer(mapxtremeServletURL);
```

渲染地图：

```
renderer.render(imageRC);
```

将渲染的地图导出到文件：

```
renderer.toFile("comp.gif");
```

MapXtremeServlet 返回所有指定图层的图像。在调用 Renderer 对象的 render() 方法时，将向 MapXtremeServlet 发送请求，随后用于在服务器上生成图像。该图像将只在调用 toFile、toStream 或 toImage 方法时返回给用户。

此外，如果配置 MapJ 取代 MapXtremeImageRenderer，单独工作以直接获取地图数据和生成图像，则将可使用 LocalRenderer 来在本地渲染图像。

无需使用渲染器，在 MapJ 客户端和 MapXtremeServlet 之间即可进行交互。但是，渲染器是地图图像返回给用户的唯一途径。用户可以创建和初始化 MapJ 对象并执行控制对象或查询地图的若干种方法，但是要查看当前地图，必须使用 **render** 方法。如果要采用多个步骤创建地图，然后再显示该地图，则上述方法尤为实用。

*第 11 章：渲染涉及的考虑因素*中对 MapXtremeImageRenderer 和 LocalRenderer 作出了进一步的讨论。

控制地图视图

在显示地图之后，可能需要更改其视图，采用更近的距离来查看地图细节，获取更宽的视图。MapJ 具有若干种用于控制地图视图的方法：setZoom()、 setCenter() 和 setZoomAndCenter()。

设置地图缩放级别

缩放级别相当于地图之上的距离。缩放级别可以更改为任意距离。所用的单位将是当前的距离单位。缩放级别最初在加载 geoset 或地图定义文件时设置。要更改地图的缩放级别，可使用 **setZoom** 方法。以下是设置缩放级别的示例：

```
// Assuming that the current distance units are
// kilometers, this command will set the map zoom to 500
// kilometers.
myMap.setZoom(500);
```

地图重定中心

设置地图的中心是控制地图视图的必要步骤之一。使用地图时，可能需要将地图中心定位于一个特定的位置或坐标。此时可以借助于 **setCenter** 方法。为此必须将 DoublePoint 传递到 **setCenter** 方法。DoublePoint 由一对 XY 坐标定义。

如果点的位置是通过用户点击地图上特定位置生成的，则返回时通常采用像素为单位表示。MapJ 需要采用以数字坐标表示的位置，为此需要使用转换方法 **transformScreenToNumeric**。MapXtreme Java 使用数字坐标系作为内部计算的基础坐标系。

以下示例展示创建屏幕点，并将其转换为 “现实世界”的点和设置地图中心的完整过程。DoublePoint 是一个通过 x 和 y 坐标定义的点。

```
// Create the screen point
screenpoint = new DoublePoint(event.getX(),event.getY());
// Create the real world point
worldpoint = myMap.transformScreenToNumeric(screenpoint);
// Set the center of the map
myMap.setCenter(worldpoint);
```

设置缩放级别和地图中心

setZoomAndCenter 方法设置当前缩放级别和地图的中心。缩放级别采用距离单位来设置。地图中心使用数字坐标系。以下是设置缩放级别和地图中心的示例。

```
// Create the screen point
screenpoint = new DoublePoint(event.getX(),event.getY());
// Create the real world point
worldpoint = myMap.transformScreenToNumeric(screenpoint);
// Set the zoom to be twice the current zoom
// and center on the point where the user clicked
myMap.setZoomAndCenter(currentZoom * 2, worldpoint);
```

设置地图边界

使用 **setBounds** 可设置地图的边界矩形。该方法取用的 DoubleRect 由表示两个对角或其中点、宽度和高度的坐标定义。后续内容对此作出了进一步的说明。

本例使用对角来设置放大地图区域的边界。

```
DoubleRect bounds = new DoubleRect(
    -1.969272, 50.560410, 1.443363, 52.315529);
```

本例使用中点、宽度和高度来设置世界地图的边界。

```
DoubleRect bounds = new DoubleRect(new DoublePoint(0,0),360,180);
myMap.setBounds(bounds);
```

设置坐标系

坐标系数据存储在名为 micsys.txt 的文件中，该文件位于 MapXtreme-4.7.0/lib/common 目录下的 micsys.jar 中。micysys.txt 文件列出了数以百计的支持坐标系以及相关的定义参数。

坐标系可通过 MapJ 方法 **setDisplayCoordSys** 来设置。

```
String csProj = new String(
    "\"Azimuthal Equidistant (North Pole)\", 5, 62, 7, 0, 90, 90");
CoordSys ts = CoordSys.createFromPRJ(csProj);
myMap.setDisplayCoordSys(ts);
```

此外，还可以使用 **createFromMapBasic** 来读取 MapBasic 字符串并通过某些预定义的常数，来设置坐标系。

有关详细信息，请参阅 HTML API 参考资料中的 CoordSys 类。

设置地图距离单位

地图单位可通过 MapJ 方法 **setDistanceUnits** 来设置。

```
LinearUnit distUnit = LinearUnit.kilometer;
myMap.setDistanceUnits(distUnit);
```

添加图层

对于向地图添加额外的数据而言， Layers 对象的 **addLayer** 方法是最常用的 MapJ API 方法之一。尽管 addLayer 方法简单且便于调用，但是还是需要采用若干步骤来说明所要添加的数据、取用数据的位置和方法。数据可以采用文件形式或数据库记录形式，从本地或远程数据源取用。MapXtreme Java 使用数据提供方管理相应操作，有关说明请参阅*第 10 章：在图层中绘制地图* 。

要访问 Layers 对象，可使用 MapJ 的 **getLayers** 方法。

通过编程保存地图

地图可以通过编程保存以供日后使用。此时既可选择存储为文件或数据库中的地图定义，也可选择存储为命名地图，后者为地图的图层集合赋予唯一的名称。

将地图定义保存到文件

要将地图定义保存到文件，可象在加载地图定义时一样创建一个 FileMapDefContainer 容器，然后调用 saveMapDefinition 方法，并向其传递所创建的容器和名称。

将地图定义存储到数据库表

在远程数据库中，地图定义采用长文本字符串 （采用 XML 格式）的形式，存储为在远程数据库的任意表的任意的 Character Large Object (CLOB) 类型字段中。MapXtreme Java 可以在大小足以存储地图定义的任意列中存储地图定义。在 VARCHAR 列中，存储的地图定义可能非常有限。列必须可以存储地图定义大小的字符串。CLOB 列可以存储 2 或 4Gb 的数据，具体大小取决于数据库类型。所有主要的数据库均具有 CLOB 类型列或与其非常类似的列。

MapXtreme Java 需要任意可存储地图定义的表至少具备以下条件：

1. 用于存储地图定义名称的 CHAR、 VARCHAR 或类似类型的列。
2. 可以存储实际 XML 文本的 CLOB 列。

当然，除了上述列出的类型之外，还有其他众多的列可用。

以下示例展示了创建名为 MAPDEFINITIONS 表的 CREATE TABLE 语句 （当前用户为该表的所有者）：

```
CREATE TABLE MAPDEFINITIONS (NAME VARCHAR(40), MAPDEF CLOB)
```

执行此语句将在远程数据库中创建 MAPDEFINITIONS 表。本语句指定 NAME 列存储最多 40 个字符的字符串，指定 MAPDEF 列存储 CLOB （打文本）值。

MapXtreme Java 安装程序安装的示例 SQL 脚本可用于在远程数据库中创建表 MAPDEFINITIONS，该表的所有者为 MAPINFO （示例位于 /MapXtreme-4.7.0/examples/server/scripts 目录中）。这些脚本执行的 CREATE TABLE 语句与此前所述语句非常相似，但是其创建的表 MAPDEFINITIONS 有所不同，该表具备以下荐用结构：

列	类型
NAME	VARCHAR(40)
MAPDEF	CLOB

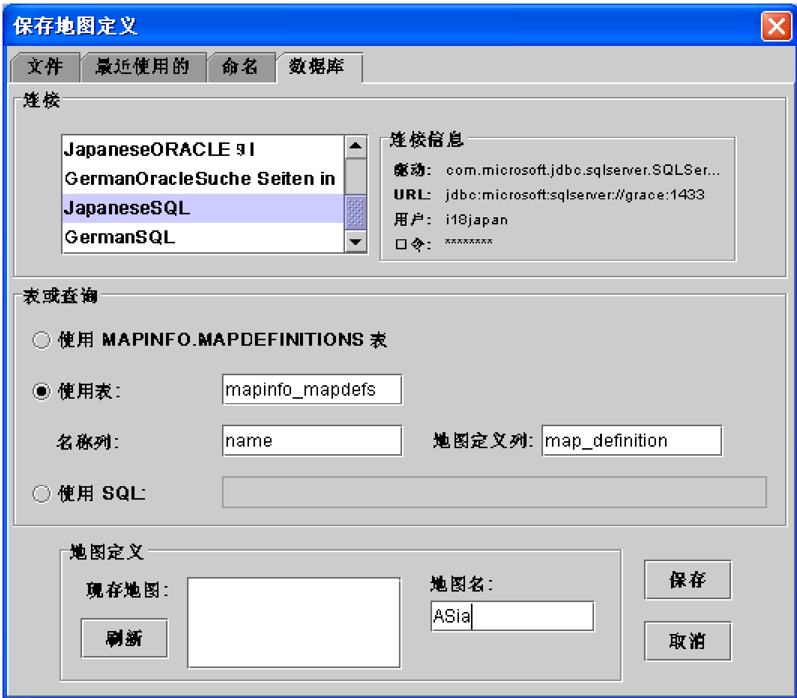
MapDefContainer 接口

通过编程保存地图定义调用了 MapDefContainer 接口，该接口提供了三个容器实施，具体取决于信息存储的方式：

- FileMapDefContainer 将地图定义存储到文件。
- JDBCMapDefContainer 将地图定义存储到数据库。
- FileMapDefContainer 将地图定义存储到命名资源库。

此外，还有两个与特定数据库相关的实施用于 JDBCMapDefContainer：OraSoMapDefContainer for Oracle 和 IUSMapDefContainer for Informix。所有其他 JDBC 数据库 （如 SQL Server）均应该使用 JDBCMapDefContainer。有关命名地图的详细信息，请参阅第 162 页。

MapXtreme Java 管理器中的 MapDefChooser 对话框将提示用户指定如何构建 FileMapDefContainer、 JDBCMapDefContainer 或 NamedMapDefContainer，如下所示：



要将 XML 地图定义写入到 MapDefContainer，可使用 MapJ 类中的 **saveMapDefinition()** 方法。指定要将地图定义保存到的容器的名称，以及所要保存的地图的名称。

以下代码展示如何将地图定义保存到 MAPINFO.MAPINFO_MAPDEFS 表。

```
String driverName = "oracle.jdbc.driver.OracleDriver";
String dbUrl = "jdbc:oracle:thin:@machinename:1521:dbSID";
String usrName = "username";
String pwd = "password";
OraSoMapDefContainer mdc = new OraSoMapDefContainer(
    driverName, dbUrl, usrName, pwd);
map = new MapJ();
map.loadGeoset("c:\\maps\\asia.gst", "c:\\maps", null);
map.saveMapDefinition(mdc, "Asia");
```

命名地图

为了便于检索保存的地图，可能需要通过唯一的名称来调用地图。本节介绍如何将地图保存为命名地图并通过编程来检索相应的地图。要通过 MapXtreme Java 管理器命名资源面板来管理命名地图，请参阅*第 5 章：管理 MapXtreme Java* 。

使用 NamedMapDefContainer 存储命名地图

要通过编程保存命名地图，所用的 MapJ.saveMapDefinition() 版本必须可以接受 MapDefContainer 和地图名称。对命名地图，需要创建 NamedMapDefContainer。要创建 NamedMapDefContainer，需要使用 JNDI Context (javax.naming.Context)。这既可以是 InitialContext (javax.naming.InitialContext)，也可以是 InitialContext （或通过 lookup() 经由 InitialContext）的子上下文。有关上下文和 InitialContexts 的详细信息，请参阅 Javadoc 中有关 JNDI API 的介绍。

要创建初始上下文，需要知道提供方的 URL，该 URL 是 （最可能是） NamedResourceServlet 的 URL。然后调用 NamedResourceContextFactory 类的 createInitialContext(providerURL) 工厂方法，如下所示：

```
Context initCtx = NamedResourceContextFactory.createInitialContext
    ("http://torpedo:8080/mapxtreme47/namedresource");
```

此时可以使用 InitialContext 来创建 NamedMapDefContainer：

```
NamedMapDefContainer container = new NamedMapDefContainer(initCtx);
```

现在需要决定在何处存储命名图层，该位置应该是相对于命名资源库根的位置。（切记 NamedResourceServlet 已知命名资源库的根的所在位置）。

例如，在名为 mymaps 的库的根下创建目录之后，需要将 MapJ 的当前状态作为 "my map" 保存到该子目录。这一操作如下所示：

```
// mapJ was previously initialized
mapJ.saveMapDefinition(container, "mymaps/my map");
```

本例中， "mymaps/my map" 表示一个复合名称。在指定复合名称时，名称的每个组成部分均必须以 / （正斜线）间隔。

注： 建议始终将命名资源 （图层）保存在库的根的子目录中。

请求命名地图的图像

在将命名地图存储在命名资源库中之后，可以从 MapXtremeServlet 请求图像。为此，可使用接收地图名称而非整个 MapJ 的 ImageRequestComposer 构造器。

```
// bounds represents the desired bounds for the resulting map image
ImageRequestComposer imgReq = ImageRequestComposer.create(
    providerUrl, "mymaps/my map", mapCoordSys, bounds, 640, 480, 256,
    Color.white, "image/gif");
```

相应代码将创建一个对 640x480 的 256 色的白色背景 GIF 的请求。

要将此图像请求发送到 MapXtremeServlet，可执行以下操作：

```
// create the remote renderer
MapXtremeImageRenderer renderer = new MapXtremeImageRenderer(
    "http://torpedo:8080/ mapxtreme47/mapxtreme");
// render the map
renderer.render(imgReq);
// export to a file
renderer.toFile("c:/temp/my map.gif");
```

基本地图之外的其他特性

MapXtreme Java 提供的 API 可以实现对地图的全方位控制。本节摘要介绍其中的主要特性。

图元

记录是一组相关的信息列。例如，客户数据库中对于每个客户均有一条记录，其中包含了姓名、地址、爱好等列。图元只是一条包含制表数据和几何信息的记录。

例如， MapXtreme 示例数据中的 World.tab 文件就是一个 MapInfo 格式数据库。对于数据库中的每个国家，均有一条记录。每个记录包括若干制表数据列，以及一个指向描述每个国家形状和位置的地理信息的引用。这样一条记录即可显示在地图之上。制表数据也称为属性数据，而几何数据也称为几何对象。这两种类型的数据构成图元。

图元并不与 MapJ 对象直接相关，但是出于若干原因，图元还是相当重要的。正如此前所述， MapJ 是程序中所有地图函数的基础。它位于对象图的顶部。

Feature 对象位于程序的最底层，用于处理特定信息。该对象是对象模型中最为明确的对象，并且与记录级别的信息相关。在 Feature 级别，可以为图形对象赋予不同的显示特性。指定图像外观的特性是由 Rendition 对象设置的。有关 Feature 对象的详细信息，请参阅 *第 13 章：图元和搜索* 。

样式

每个图元均具备与其相关的样式，用于描述其在地图上的外观。样式属性可分为如下 3 个一般类别：填充、单笔和符号。填充属性控制区域的填充方式。单笔属性控制线条 （或线几何对象或区域的边缘）的绘制方式。符号属性控制如何绘制用于点几何对象、线标记或符号填充的符号。

控制样式的部分 MapJ API 是 Rendition 类。由于有众多方法可用，因此组合使用可用样式的方式实际上没有任何限制。样式可以指定给图元、图层、标注和专题，并可用于覆盖象征符号。有关样式的详细信息，请参阅*第 14 章：标注和样式*。

专题

无论是通过图层的 **addLayer** 方法、MapJ **loadGeoset** 方法，还是通过 **loadMapDefinition** 方法添加图层，每个图层均具有其自己的特征，例如线条颜色、线条宽度等。这些特征是以数据源中的信息为基础的。通常相应设置对于整个图层而言是一致的。例如，如果要从示例数据加载 World.tab，每个国家均将显示为一个由绿色填充的实体图案。图层中的每个图元显示的方式均相同。

专题可通过编程，基于某些条件来更改部分或全部图层中的图元外观。例如，如果要更改人口超过 5 千万的所有国家的颜色，可以通过使用专题来实现。此处提供了 4 种可用的 Theme 类：

- OverrideTheme － 更改整个图层的样式
- RangedTheme － 划分数据范围，并根据范围值来确定影线表示
- IndividualValueTheme － 通过影线表示共享特定属性值的图元组
- SelectionTheme － 对用户定义的选定图元列表应用样式

各个专题中的图元均具有与其关联的样式。Rendition 对象封装了用于图形和文本显示的样式属性。

专题还可创建用于标注，以区分同一图层中的标注。例如，创建范围标注专题用于显示和增长率相关的定制大小的标注，此时即可令较大的标注表示较高的增长率。有关专题地图绘制的详细信息，请参阅*第 15 章：专题地图绘制和分析*。

10

在图层中绘制地图

本章介绍表和地图之间的关系以及两者如何通过图层创建所需级别的细节。

本章内容：

作为图层的地图

我们此前已经介绍了作为图层集合的计算机地图概念。每个包含图形对象的表均可显示为地图图像中的图层。

可将这些图层视为透明，其中每个图层包含地图的不同组成部分。图层上下叠加在一起即可展现地图的全貌。例如，第一个图层包含国界，第二个图层包含表示首都的符号，第三个包括高速公路。这些透明图层上下叠加，即可构成一个完整的地图。

图层由地理图元和关联数据构成。例如，国界图层中的区域定义了每个国家的边界，其中可能具有表示各国人口、文化程度或平均家庭收入的属性。通过使用具有附加信息的图层来创建地图，除了可以创建精美的地图 , 还可查询图层信息进行分析和显示。此类地图在显示地图数据之间的关系时尤为高效。

本章侧重于如何通过程序处理图层，例如定义图层和向地图添加新图层。本章还介绍包括注释图层 （Annotation 图层）和栅格图像在内的特殊类型的图层。要了解如何使用 MapXtreme Java 管理器的地图定义 GUI 加载图层，请参阅*第 5 章：管理 MapXtreme Java* 。

Layers 集合：构建地图的模块

Layers 集合包含 Layer 对象，可从 MapJ 访问。Layer 对象构建自表并用于构成地图。每个图层包含不同的地图图元，例如区域、点或线。Layers 集合具有用于执行从集合增减 Layer 对象等操作的方法。Layer 对象具有的搜索方法可用于确定图层上的特定信息。

Layer 对象表示构成地图图元的数据，这些地图图元通常具有突出的图元类型，例如区域、线或符号。通常 Layer 对象和源自一个表的地理对象相对应。Layers 集合中的每个 Layer 对象的行为都相互独立。其样式、缩放图层均可在单独更改，而不会影响到任何其他图层。

Layer 对象利用若干个相关的类，例如 ThemeList、 LabelProperties、 FeatureSets、 ColumnStatistics 和 TableInfo。除了用于访问这些对象的方法之外， Layer 对象的搜索方法还可以用于确定图层上的特定信息。通过 Layer 对象即可以利用 MapXtreme 的大部分功能。

如何构建 Layers 集合

要构建地图，可以从向 Layers 集合添加图层入手。在前一章，我们介绍了使用 **MapJ.loadMapDefinition** 和 **MapJ.loadGeoset** 加载地图数据的代码。 Layers 集合是通过相应地图定义的图层来确定的。在加载地图定义之后，显示坐标系将会更新，并删除此前的任何图层。

在创建 Layers 集合之后，即可添加更多图层。使用 **Layers.addMapDefinition** 方法时，相应图层将添加到当前地图，维持现有的坐标系设置。图层也可使用 **Layers.addLayer** 方法单独添加。 addLayer 方法将该图层置于集合的末端。使用 insertLayer 可控制相应的位置。

如何绘制图层

Layers 集合中的地图图层将按照从 0 开始的索引以升序显示。 Layer(0) 是顶层的图层，Layer(1) 是 Layer(0) 下面的图层，依此类推，底层图层最先绘制，顶层图层最后绘制。保持正确的图层顺序很重要。

例如，当前有一个客户地点图层和一个人口普查地点图层。如果图层在 Layers 集合中的排序不正确， MapXtreme 将最先绘制客户点，然后再显示人口普查地点图层。这样客户地点将由于人口普查地点图层的存在而变得模糊。有关解释图层定位的代码示例，请参阅第 184 页的 *Layers 集合的方法* 。

使用数据提供方定义图层

要将图层添加到 Layers 集合，必须先对其进行定义。数据提供方是定义图层的关键所在。

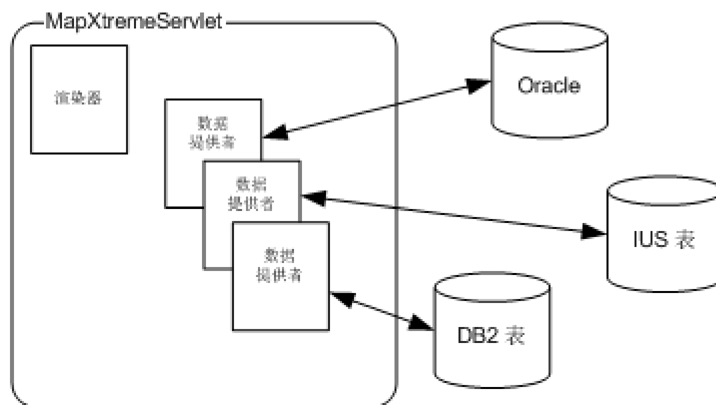
每个图层均有一个内部对象用于负责数据访问，这个对象就是数据提供方。数据提供方不是由用户直接创建的，但是其说明定义了图层。以下三个接口用于说明数据提供方 （进而说明图层）：

- TableDescHelper – 说明数据
- DataProviderHelper – 定义数据源
- DataProviderRef – 说明如何获取数据

MapXtreme Java 的 DataProvider 可用于为以下数据源创建地图图层：

- MapInfo 制表格式 (.tab)
- 有空间选项的 Oracle
- Informix Universal Server SpatialWare DataBlade
- SQL Server for SpatialWare

- JDBC 兼容的表，相应的表中包含经度和纬度列
- GeoTIFF 和 MIGrid 栅格文件
- ESRI Shapefiles
- 数据绑定[1]
- 注释[2]



TableDescHelpers

TableDescHelper 是一个用于说明所要访问的数据的接口。对于 MapXtreme 可以访问的每种不同类型的数据源，都有一个 TableDescHelper 与之相对应。并且都具有一个和相应数据源相关的构造器参数。

例如，用于说明 MapInfo 表 world.tab 的 TABTableDescHelper 只需要一个表名来进行说明。用于说明 Oracle 数据的 OraSoTableDescHelper 通过表名或 SQL 查询定义。本章的后续内容提供了相应的代码示例。有关每个 TableDescHelper 的更多详细信息，可见于联机 Javadoc 参考资料。

DataProviderHelpers

DataProviderHelpers 定义数据源。如果是 MapInfo TAB 文件，则包含一个 .tab 文件的目录就是该文件的数据源。因此，用于 tab 文件的 DataProviderHelper TABDataProviderHelper，将目录作为其唯一的参数。

1. 可用于从 .tab 和 JDBC 数据源接检索数据，以便显示为单个图层。

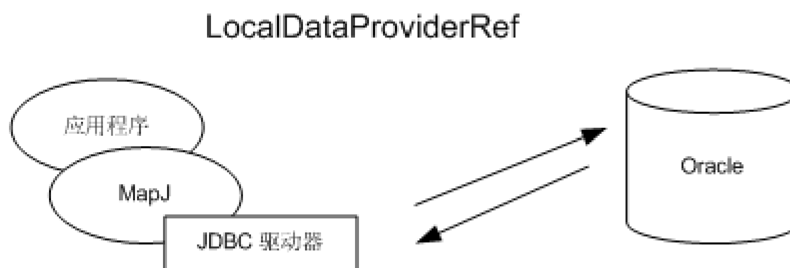
2. 不是一个典型的数据提供方，其原因在于信息并未存储在数据库中，而是保存在内存中。有关详细信息，请参阅第 180 页的*注释图层（Annotation 图层）*。

如果是由若干个表构成的地图，并且这些表都保存在 Oracle 数据库的一个实例中。那么此数据库将成为每个表的数据源。 OraSoDataProviderHelper 取用说明数据源的参数，例如服务器主机名、服务器端口号、用户名和口令等。同一 DataProviderHelper 可用于源自同一数据源的不同的表。有关每个 DataProviderHelper 的更多详细信息，可参阅联机 Javadoc 参考资料。

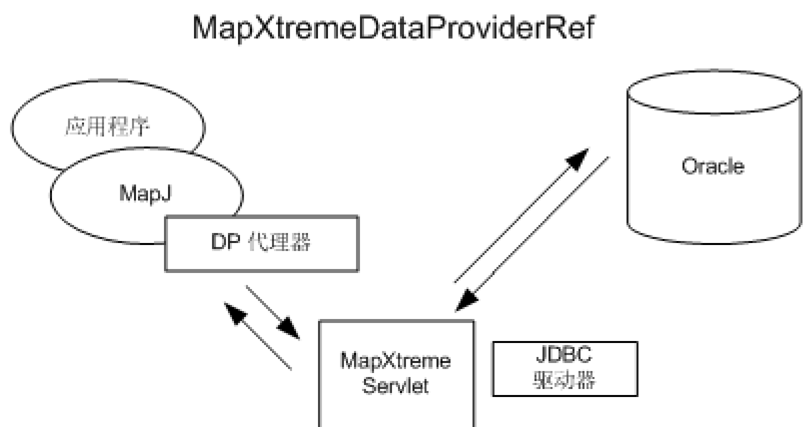
DataProviderRef

数据提供方内置远程操作的功能。 DataProviderRef 说明了用于访问数据源的对象。此处有两种可能：1) 应用程序 （进程）包含 MapJ 且此图层可以直接访问数据源，或者 2) 应用程序可以通过 MapXtremeServlet 来间接访问数据源，然后由 MapXtremeServlet 将数据传输回给应用程序。直接的方法是通过 LocalDataProviderRef 来执行，间接的方法使用 MapXtremeDataProviderRef。

使用 **LocalDataProviderRef** 表示应用程序将直接与数据源通信。这意味着该应用程序必须可以使用访问数据源所需的任意资源。例如，JDBC 驱动程序必须位于应用程序的 classpath 中，以便用于直接访问 RDBMS 中的数据。



客户机借助于 MapXtremeServlet 获取数据时将会使用 **MapXtremeDataProviderRef**。在使用三层部署的体系结构时，这是最为典型的情况。此时，客户机只存在一个 “存根”数据提供方，而实际提供方将通过 MapXtremeServlet 创建，并在随后从数据源访问数据。采用这一部署，访问数据源所需的资源仅限于中间层。这样无需在客户机内部署 JDBC 驱动程序， MapXtremeServlet 即可访问 RDBMS 中的数据。



何时访问数据？

TableDescHelpers、 DataProviderRefs 和 DataProviderHelpers 用于定义 DataProvider。在最先定义和创建图层时，并没有发生数据访问。作为类似对于图层的 getTableInfo 的调用请求、图层搜索方法请求或渲染请求等特定请求的响应， DataProviders 只访问其底层数据源。

创建定制数据提供方

MapXtreme Java 还支持定制的数据提供方。有关如何构建定制数据提供方的详细信息，请参阅*第 18 章：创建定制数据提供方*。

定义 JDBC 图层的注意事项

每个 JDBC 数据源都具有若干个相应的 DataProviderHelper 构造器。尽管有些构造器使用起来比较方便，但是我们强烈建议设置 JDBC 图层使用连接池。这需要使用最一般形式的 DataProviderHelper 构造器。有关连接器的详细信息，请参阅*第 12 章：访问远程数据*。另请参阅位于任意 JDBC DataProviderHeleper 下的 Javadoc 所含代码实例。

用于 JDBC 图层的 TableDescHelper 对象均具有某些共同的参数。当 JDBC 图层在定义为表而不是 SQL 语句时，这些参数中的部分参数将成为可选。其中包括空间列、几何坐标系、几何尺寸和表级别的样式。如果这些参数均未出现，则将搜索

MapInfo.MAPINFO_MAPCATALOG 查找相应信息。如果已知相应的值，我们强烈建议在构建该对象时提供相应信息。以避免对 MAPCATALOG 进行额外的查询，从而提高应用程序的性能。

在没有指定主键列的情况下， MapXtreme 将在数据库中查询表的方案定义，查找适合于用作键的列。首先将尝试找到正式定义为表的主键列的列；如果没有主键列， MapXtreme 将选择一个数据库可确保其对于每行都唯一的列，该列不是虚列，且其类型应该为字符或数字。在 TableDescHelper 中确认主键列避免了因上述操作所耗费的资源，同时确保使用主键（如在选择类中）时出现预期的行为。此外，在向 JDBC 列添加图元时，还将会对主键列进行特殊处理 （请参阅第 241 页的*注释图层图元的主键*），因此在此指定主键列十分重要。

JDBC 图层支持每图元的样式，即可为每个存储在数据库中的图元提供单独的样式。用于数据库表的样式列包括在 MAPINFO_MAPCATALOG 中，位于名为 RENDITIONCOLUMN 的列之下。如果使用 EasyLoader 将 TAB 文件上载到数据库，系统即会创建此列。 MapInfo 网站上提供了 EasyLoader 供您使用。

此外也可以为 JDBC 图层指定每图元标注样式。以便可为单个图层创建不同的样式标注。实际的样式存储在定义图层的表的列中。但在这一情况下， EasyLoader 和 MAPCATALOG 不支持每图元标注样式。此时必须提供列，并在创建用于数据源的 TableDescHelper 键入。

将图层添加到 MapJ Layers 集合

这是添加图层的常规步骤。

1. 创建 TableDescHelper
2. 创建 DataProviderHelper
3. 创建 DataProviderRef（需要 DataProviderHelper 作为输出）
4. 使用 Layers.**addLayer** 方法 （取 DataProviderRef 和 TableDescHelper 作为输入）。默认情况下，这将该图层置于集合的底部。此外还可以使用 **Layers.insertLayer**。

各类 MapXtreme Java 支持的数据源都具有 TableDescHelper 和 DataProviderHelper 实施。相关摘要如下表所示。有关详细信息，请参阅 Javadoc 中的 HTML 参考资料。

数据源	**TableDescHelper**	**DataProviderHelper**
MapInfo 表	TABTableDescHelper	TABDataProviderHelper
Oracle with Spatial Option	OraSoTableDescHelper	OraSoDataProviderHelper
Informix Universal Server SpatialWare DataBlade	IusSpwTableDescHelper	IusSpwDataProviderHelper

数据源	TableDescHelper	DataProviderHelper
SQL Server for SpatialWare	SQLServerSpwTableDescHelper	SQLServerSpwDataProvider Helper
JDBC 兼容的表，该表中包含经度列和纬度列	XYTableDescHelper	XYDataProviderHelper
注释图层 （Annotation 图层）	AnnotationTableDescHelper	AnnotationDataProviderHelper
GeoTIFF 栅格	GeoTIFFTableDescHelper	GeoTIFFDataProviderHelper
ESRI Shapefiles	ShapeTableDescHelper	ShapeDataProviderHelper
数据绑定 *	DataBindingTableDescHelper	DataBindingDataProviderHelper
MapInfo 网格	MIGridTableDescHelper	MIGridDataProviderHelper
Northwood 网格	NWGridTableDescHelper	NWGridDataProviderHelper

* 用于绑定源自同一 MapJ 图层中的 .tab 和 JDBC 数据源数据的特殊数据提供方。有关详细信息，请参阅第 178 页的*数据绑定*。

用于 TAB、 Oracle 和 JDBC 兼容的数据源的代码示例如下。联机 Javadocs 参考资料中也提供了这些示例和其他相关示例。

TAB 数据提供方示例

以下是创建 TABDataProvider 并将图层指定给 MapJ 的示例。

```
// specify the url to the MapXtreme servlet which remotely connects us
// to the map engine
String mapXtremeURL = "http://stockholm:8080/mapxtreme47/mapxtreme";

// create the tab TableDescHelper
TABTableDescHelper tabTDHelper = new TABTableDescHelper(
    new File("mytab.tab").getName());

// create the tab DataProviderHelper
TABDataProviderHelper tabDPHelper =
    new TABDataProviderHelper("d:\\maps");

// Create the Remote DataProviderRef needed to access the Data
MapXtremeDataProviderRef mxtDPRef = new
MapXtremeDataProviderRef(tabDPHelper, mapXtremeURL);

// assign it to MapJ
map.getLayers().addLayer(mxtDPRef, tabTDHelper, "tabLayer");
```

Oracle 数据提供方示例

以下是创建 OracleDataProvider 并将图层指定给 MapJ 的示例。务必确保 JDBC 驱动程序位于 classpath 中。

```
// Specify the URL to the MapXtreme servlet that will be used to access
// the data
String mapXtremeURL = "http://stockholm:8080/mapxtreme47/mapxtreme";

// Create the DataProviderHelper
// Using pooled connections (Recommended)
CommonDataProviderHelpers

// Using Database specific DataProviderHelper
OraSoDataProviderHelper oraDPHelper=new
OraSoDataProviderHelper("dbName",1521, "dbSid", "mary", "mary123",
DriverType.thin, "oracle.jdbc.driver.OracleDriver");

// Create a String array with the name(s) of the column(s) to use as a
unique key for records in the table
String[] idColumn = {"mi_prinx"};

// Create a TableDescHelper
// Required constructor when using a tablename
OraSoTableDescHelper oraTDHelper = new OraSoTableDescHelper(
    "states", false, idColumn, "geoloc", null, RenditionType.none,
    null, RenditionType.none, CoordSys.longLatWGS84, 2, "mary");

// Required constructor when using a query
OraSoTableDescHelper oraTDHelper = new OraSoTableDescHelper(
    "select pop_1980, pop_1990, state_name, geoloc, mi_prinx from
    states where pop_1990 < pop_1980 * 1.03", idColumn, "geoloc",
null,
    RenditionType.none, null, RenditionType.none,
    CoordSys.longLatWGS84, 2);

// Reference the remote DataProvider needed to access the data
MapXtremeDataProviderRef mxtDPRef = new MapXtremeDataProviderRef(
    oraSoDPHelper, mapXtremeURL);

//Add the layer (assume mapj is a MapJ object)
m_myMap.getLayers().addLayer(
    mxtDPRef, oraTDHelper, "Oracle Spatial Layer");
```

XY Data Provider Example

这一代码示例创建用于 JDBC 数据源的数据提供方，此时空间信息存储在 X, Y 列中。务必确保 JDBC 驱动程序位于 classpath 中。

```
// Specify the url to the MapXtreme servlet that will be used to access
// the data
String mapXtremeURL = "http://localhost:8080/mapxtreme47/mapxtreme";

// Create the DataProviderHelper
// Using pooled connections (Recommended)
// Using database specific DataProviderHelper (in this case the XY data
// is located in an Oracle database)
XYDataProviderHelper xyDPHelper = new XYDataProviderHelper(
    "oracle.jdbc.driver.OracleDriver",
    "jdbc:oracle:thin:@serverName:1521:dbSid", "mary", "mary123");

// Create a String array with the name(s) of the column(s) to use as a
// unique key for records in the table
String[] idColumn = {"city_name"};

// Create a TableDescHelper
// Required constructor when using a tablename:
XYTableDescHelper xyTDHelper = new XYTableDescHelper(
    "city125", "mary", false, "longitude", "latitude", null,
    RenditionType.none, null , RenditionType.none , idColumn,
    CoordSys.longLatWGS84);

// Reqiured constructor when using a query:
XYTableDescHelper xyTDHelper = new XYTableDescHelper(
    "select longitude, latitude, city_name from city125 where pop_1990
    > 50000", idColumn, "longitude","latitude", null,
    RenditionType.none, null ,RenditionType.none,
    CoordSys.longLatWGS84);

// Reference the remote DataProvider needed to access the data
MapXtremeDataProviderRef mxtDPRef = new MapXtremeDataProviderRef(
    xyDPHelper, mapXtremeURL);

// Add the layer (assume mapj is a MapJ object)
mapJ.getLayers().addLayer(mxtDPRef, xyTDHelper, "XY Layer");
```

SQL Server Data Provider Example

以下是创建 SQL Server DataProvider 并将图层指定给 MapJ 的示例。务必确保 JDBC 驱动程序位于 classpath 中。

```
// Specify the url to the MapXtreme servlet
String mapXtremeURL = "http://localhost:8080/mapxtreme47/mapxtreme";

// Create the DataProviderHelper
// Using pooled connections (Recommended)
// Using database-specific DataProviderHelper
SQLServerSpwDataProviderHelper sqlDPHelper = new
    SQLServerSpwDataProviderHelper("machineName",1526,"mary",
    "mary123","com.merant.datadirect.jdbc.sqlserver.SQLServerDriver");

// Create a String array with the name(s) of the column(s) to use as a
// unique key for records in the table
String[] idColumn = {"sw_member"};

// Create a TableDescHelper
// Required constructor when using a tablename:
SQLServerSpwTableDescHelper sqlTDHelper = new
    SQLServerSpwTableDescHelper("states", "mary", false, idColumn,
    "sw_geometry", null, RenditionType.none, null, RenditionType.none,
    CoordSys.longLatWGS84);

// Required constructor when using a query:
SQLServerSpwTableDescHelper sqlTDHelper = new
    SQLServerSpwTableDescHelper("select state, statecap, sw_member,
    sw_geometry from states where pop_1990 > 2000000", idColumn,
    "sw_geometry", null, RenditionType.none, null, RenditionType.none,
    CoordSys.longLatWGS84);

// Reference the remote DataProvider
MapXtremeDataProviderRef mxtDPRef = new MapXtremeDataProviderRef(
    sqlDPHelper, mapXtremeURL);

// Add the layer (assume mapj is a MapJ object)
mapj.getLayers().addLayer(mxtDPRef, sqlTDHelper, "SQLServer Layer");
```

数据绑定

MapXtreme Java 支持数据绑定，其中的 MapInfo .tab 文件可以与 JDBC 数据源合并为 MapJ 之中的单一图层。使用数据绑定访问 .tab 文件中的几何数据，将其与源自缺少空间信息的数据库记录的属性数据相结合。

数据绑定通过位于 com.mapinfo.dp.databinding 文件包之内的数据提供方处理。

要通过编程合并两个表，可执行以下操作：

1. 为每个表创建 TableDescHelper。
2. 创建 DataBindingTableDescHelper，令其构建器取用在上一步中创建的两个 TableDescHelpers。此外，需要链接两个表的列名。列名无需相同。一个或多个列均可用于绑定两个数据源。
3. 为每个表创建 DataProviderHelper。
4. 创建 DataBindingDataProviderHelper，令其构建器取用在上一步中创建的两个 DataProviderHelper。
5. 创建在 DataBindingDataProviderHelper 中传递的 DataProviderRef。
6. 将图层添加到 Layers 集合。

DataBindingDataProviderHelper 将 searchXXX 调用分配给适当的 DataProvider，并同步两个结果。

此外，还可向 VisualMapJ 通过 Layer Control bean 来添加数据绑定图层。

有关在增加图层向导中添加数据绑定图层的详细信息，请参阅*第 5 章：管理 MapXtreme Java* 中的 “添加数据绑定图层” 一节。

数据聚合

数据聚合是多个记录 / 图元作为两个数据源关联结果合并的过程。

MapXtreme Java 包含一个类 com.mapinfo.dp.databinding.Aggregation，该类定义属性源的列值如何聚合到数据绑定图层。例如，对于以下两个表：

- 一个带有区域名属性列且包含销售区域的几何 .tab 文件。
- 一个包含区域名和每个销售员销售额的属性数据库。其中每个销售区域有多个销售员。

现在如果对点执行搜索，则将会返回销售区域 A，该区域有两个销售员，相应的 FeatureSet 中将返回两个图元。以下是可能的结果：

区域名称	销售额	销售员
A	100000	Smith
A	200000	Peterson

现在定义一个聚合，或是说总计。代码如下所示：

```
dbDesc.addAggregation(Aggregation.SUM);
```

如果执行相同的搜索，我们现在将只能获取一个图元。以下是可能的结果：

区域名称	销售额
A	300000

MapXtreme Java 将采用以下方式聚合数据：

- 总计 — 返回该组中所有记录值的总和
- 计数 — 返回一组记录中的记录数
- 均值 — 返回该组中所有记录值的平均值
- 最小值 — 返回该组所有记录中最小的值 （或非数字列中的第一个值）
- 最大值 — 返回该组所有记录中最大的值 （或非数字列中的最后一个值）

用于非数字列的有效聚合为最小值、最大值和计数。

可以为每个列定义一个聚合。也可以根据需要删除聚合。

如果执行搜索，并要求返回 columnA 和 columnB，但是只定义了用于 columnB 的聚合，此时将会报错。或者所有返回的列必须定义聚集，或者所有的列都不定义聚集。

所有和聚集一起使用的方法均位于 DataBindingTableDescHelper 上。

通过特定搜索返回的记录包括所有几何信息，即使在数据库 （属性表）中没有相应记录。

代码示例：数据绑定

```
String tabDir = "c:\\maps\\";
TABDataProviderHelper tabDPH = new TABDataProviderHelper(tabDir);
String tabFile = "states.tab";
TABTableDescHelper tabDesc = new TABTableDescHelper(tabFile);
String dbUrl = "jdbc:oracle:thin:@spatial.mapinfo.com:1521:maps";
String user = "spatial";
String password = "spatial123";
String driverClass = "oracle.jdbc.driver.OracleDriver";
```

```
JDBCDataProviderHelper jdbcDPH = new JDBCDataProviderHelper(
    dbUrl, user, password, driverClass);
String tableName = "USA";
JDBCTableDescHelper jdbcDesc = new JDBCTableDescHelper(
    tableName, user);
DataBindingDataProviderHelper dbDPH = new
    DataBindingDataProviderHelper(tabDPH, jdbcDPH);
String[] tabBindCol = {"State", "State_Name"};
String[] jdbcBindCol = {"STATE", "STATE_NAME"};
DataBindingTableDescHelper dbDesc = new
    DataBindingTableDescHelper(tabDesc, jdbcDesc, tabBindCol,
    jdbcBindCol);
String servletUrl = "http://serverhost:8080/mapxtreme47/mapxtreme"
MapXtremeDataProviderRef dpRef = new MapXtremeDataProviderRef(
    dbDPH, servletUrl);
mapj.getLayers().addLayer(dpRef, dbDesc, "DataBinding Layer");
```

注释图层 （Annotation 图层）

注释图层 （Annotation 图层）是特殊的地图图层，该图层包含在特定地图区域上标记或放置重点的图元。通常，从搜索方法返回的图元将添加到注释图层 （Annotation 图层）。

例如，在特定的点半径中选择和突出显示图元，可使用 **FeatureLayer.searchWithinRadius** 方法，返回该点的圆图元。要显示搜索半径，可使用 FeatureFactory 的 **createCircularRegion** 方法。在创建图元之后，可使用 **addFeature** 方法来向注释图层 （Annotation 图层）添加新图元。注释图层 （Annotation 图层）在搜索之前或搜索之后创建均可。

此外还具有多个注释图层 （Annotation 图层）。用于注释图层 （Annotation 图层）的表驻留在内存中。该表使用 AnnotationDataProviderHelper、 Annotation TableDescHelper 和 LocalDataProviderRef 创建。该表在创建之后，可以象其他图层一样处理。

注释图层 （Annotation 图层）的保存和任意其他图层的保存相同。添加到注释图层 （Annotation 图层）的任意图元将作为具有符合 GML 2.0 规格存储的几何信息的 OGC SimpleFeature，存储在地图定义中。

以下是创建注释数据提供方并将其添加到 MapJ Layers 集合的示例：

```
// create the annotation table desc helper
AnnotationTableDescHelper annTDHelper = new AnnotationTableDescHelper(
    "annLayer");

// create the annotation data provider
```

```
AnnotationDataProviderHelper annDPHelper =
    new AnnotationDataProviderHelper();

// An Annotation layer requires the use of local memory space, so we
//create a Local DataProvider Ref
LocalDataProviderRef localDPRef = new LocalDataProviderRef(
    annDPHelper);

//assign it to MapJ - note getLayers()
mapJ.getLayers().addLayer(localDPRef, annTDHelper, "AnnLayer");
```

分析图层

MapXtreme Java 提供了包括饼图、并行条形图和堆叠条形图的功能。并行条形图垂直显示，而堆叠条形饼图则水平显示。这些图均通过 AnalysisLayer 类表示。有关详细信息，请参阅第 288 页的*使用分析图层* 。

命名图层

命名图层是给定唯一名称的一种图层类型。命名图层和 MapXtreme Java 中的其他命名资源具有相同的优点：

- 该资源以其名称所知，而非其属性。
- 该资源驻留在一个位置，但是可从多个位置引用。
- 要更改应用程序或数据的外观或行为，只需更改资源，而非每个应用程序或数据文件。

此外，正如其他命名资源（地图、样式）一样，命名图层使用 Java 命名和目录接口 (JNDI) 应用程序编程接口 (API)。 NamedLayer 对象透明处理所有和 JNDI 的交互。

要通过 MapXtreme Java 管理器命名资源面板来管理命名图层，请参阅*第 5 章：管理 MapXtreme Java* 。

使用 NamedResource 存储命名图层

任意图层（TAB、注释、 JDBC）均可通过 JNDI API 以编程方式存储为命名图层。 JNDI 上下文 (javax.naming.Context) 包括两种用于将命名资源保存到命名资源库中的方法。如下所示：

- **bind**(String name, Object obj)
- **rebind**(String name, Object obj)

使用 bind() 方法可将全新的资源 （以前不存在）保存到库中。使用 rebind() 方法可更新库中已经存在的资源。

所需的第一个对象是 JNDI 上下文。这既可以是 InitialContext (javax.naming.InitialContext)，也可以是 InitialContext （或经由 InitialContext 通过执行 lookup() 获取）的子上下文。有关上下文和 InitialContexts 的详细信息，请参阅 Javadoc 中有关 JNDI API 的介绍。

要创建初始上下文，需要知道提供方的 URL，该 URL 是 （最可能是） NamedResourceServlet 的 URL。然后调用 NamedResourceContextFactory 类的 createInitialContext(providerURL) 工厂方法，如下所示：

```
Context initCtx =
    NamedResourceContextFactory.createInitialContext(
    "http://torpedo:8080/mapxtreme47/namedresource:);
```

现在只需要决定在何处存储命名图层，该位置是相对于命名资源库根的位置。

在名为 "my layers" 的库的根下创建目录之后，即可在 "my states" 子目录中保存特定图层。如下所示：

```
// mapJ was previously initialized
// fetch the "states" layer from the Layers collection
Layer states = mapJ.getLayers().get("states");

// create a named resource out of the layer
NamedResource resource = new NamedResource(states);

// Now save it via the container we obtained above
container.bind("my layers/my states", resource);
```

本例中， "my layers/my states" 表示一个复合名称。在指定复合名称时，每个组件名都必须以 / （正斜线）间隔。

注： 命名资源 （图层）必须始终保存在库的根的子目录中。这些资源绝对不能直接存储在根中。

将命名图层添加到 Layers 集合

要将此前存储的命名图层添加到 Layers 集合，必须使用 Layers 集合的以下方法之一：

- AbstractLayer **addNamedLayer**(String providerURL, String path, String resourceName)
- AbstractLayer **addNamedLayer**(Context context, String resourceName)
- AbstractLayer **insertNamedLayer**(String providerURL, String path, String resourceName, int pos)
- AbstractLayer **insertNamedLayer**(Context context, String resourceName, int pos)

采用 addNamedLayer() 和 insertNamedLayer() 时，接受的是 providerURL 和路径而非 JNDI 上下文，根本无需作出任何 JNDI 调用。因此，要添加位于命名资源库的根的 "my layers" 子目录中名为 "my states" 的图层，可执行以下操作：

```
// mapJ was previously initialized
Layes layers = mapJ.getLayers();
layers.addNamedLayer("http://torpedo:8080/mapxtreme47/namedresource",
    "my layers", "my states");
```

路径参数为可选 （可指定为空）。如果指定路径，那么 resourceName 参数必须相对于给定路径。如果没有给定路径，那么 resourceName 必须相对于命名资源库的根。因此可使用以上调用来替代这一 addNamedLayer() 调用，如下所示：

```
layers.addNamedLayer("http://torpedo:8080/mapxtreme47/namedresource",
    null, "my layers/my states");
```

此前接受 JNDI 上下文的 addNamedLayer() 和 insertNamedLayer() 的版本需要先获取有效的 JNDI 上下文。有关如何获取 JNDI 上下文的详细信息，请参阅以上的存储命名图层一节。

Layers 集合的方法

在添加某些图层之后，可能需要对 Layers 集合作出某些更改。本节说明有助于进行修改的若干种方法。在应用程序中，将会通过 Layers 集合来频繁引用对象和方法。

获取集合中的图层名称

本例说明的项数是集合中的图层数。例如，如果要遍历集合中的各项以获取各项的名称，这一示例将会非常有用。

```
Layers layers = myMap.getLayers();
AbstractLayer layer;
String layerName;
for (int i=0; i < layers.size(); i++)
{
    layer = layers.get(i);
    layerName = layer.getName();
}
```

从集合获取图层

get(string name) 方法从集合获取特定的 Layer 对象。**get**(string name) 方法将一个图层返回为对象。可以通过图层名称对其进行引用，例如 Highways 或 Cities。此外还可以通过其位置来对其进行引用。**get**(index at) 方法返回集合中给定位置的图层，例如 0、1、2 等。索引基于 0。以下示例展示了上述两种途径：

```
AbstractLayer myLayer;
myLayer = myMap.getLayers().get("highways");
myLayer = myMap.getLayers().get(5); //gets the 6th layer
```

插入图层

insertLayer 方法将图元图层添加到 Layers 集合，该集合提供 DataProvider 信息和放置图层的位置。类似于添加图层，插入图层时必须提供 DataProviderRef 和 TableDescHelper。集合中插入图层之后的任意图层的位置都将下移一位。

```
// inserting a layer at position 5
layers.insertLayer(dataProviderRef, tableDescHelper, 5, "newLayer");
```

移动图层

move 方法在 Layers 集合中重新定位图层。第一个参数 *From* 位置 （顶级图层 = 0），第二个参数是 *To* 位置。

```
// moving a layer from the bottom to the top
layers.move(layers.size() - 1, 0);
```

删除图层

remove 方法从地图删除指定图层。

```
//removing a layer by position (top layer)
layers.remove(0);

//removing a layer by name
layers.remove("highways");
```

删除所有图层

removeAll 方法从地图删除所有图层。

```
//removing all layers
layers.removeAll();
```

MapJ HTML 参考资料中提供了 Layers 集合方法和属性的完全列表。

检索图层的界限

使用 **getBounds()** 方法可检索图层界限，以便可以查看整个图层。

此前，检索图层的界限是一个占用大量计算机资源的操作，对于 JDBC 和栅格图层尤其如此。该操作需要查询所有图元，然后合并其界限。

在 TAB 图层上执行调用之后， getBounds() 查询 .map 文件以获取边界信息。对于 JDBC 图层， getBounds() 查询 MapCatalog （如果没有界限信息，则将返回空）。如果成功，将返回地图的数字 CoordSys 中的 DoubleRect。随后，可以将此信息用于设置图层界限的地图界限，这一方式比此前查看整个图层的效率更高。

此方法还在新的 View Entire Layer Maptool bean 中提供了底层功能。在 MapXtreme Java 管理器接口中可以见到 View Entire Layer bean 的工作示例。

代码示例

以下是使用 Layer.getBounds() 来设置 MapJ 中的一个图层或所有图层的地图界限的示例应用程序。

```
/* setting the MapJ's bounds to the DoubleRect that encompasses all of
the features in a particular layer obtain a reference to a layer in the
MapJ
*/
DataLayer lyr= (DataLayer)mapj.getLayers().elementAt(0);

//get the layer's bounds
DoubleRect lyrBounds=lyr.getBounds();

//check if the layer's bounds are null
if (lyrBounds!=null)
{
    mapj.setBounds(lyrBounds);
}

// setting the MapJ's bounds to the DoubleRect that encompasses the
// bounds of all of the layers in the MapJ

//get all of the layers in the MapJ
Layers ls= mapj.getLayers();

//create the bounds for all the layers in the MapJ
DoubleRect allLayerBounds=new DoubleRect();

//initialize it
allLayerBounds.initBounds();
for (int layerCount=0;layerCount<ls.size();layerCount++)
{

    //get the next layer
    DataLayer l =(DataLayer)ls.get(layerCount);

    //get the bounds of the layer
    DoubleRect lyrBounds=l.getBounds();

    //if the layer's bounds are not null, merge it with the bounds for
    // the entire MapJ
    if (lyrBounds!=null) {
      allLayerBounds.merge(lyrBounds); }
}

mapj.setBounds(allLayerBounds);
```

如果 getBounds() 返回为空

但如果 getBounds() 返回为空？如果获取界限信息对于您很重要，则可能需要考虑使用此前提及的 “强制” 方法，查询所有图元并合并界限信息。但是，对于较大的 JDBC 表或查询或栅格图像，此操作将会对性能造成明显影响。例如，对于不是 gif、jpeg、png 或 tiff 的栅格图像， MapXtreme Java 必须将整个图像读取到内存中，然后获取宽度和高度，为此尤其要占用大量的 CPU 时间。此时必须确定是否值得占用大量 CPU 时间来查看整个图层。

要使用强制方法获取图层边界，可执行以下代码示例：

```
DoubleRect layerBounds = new DoubleRect();
Feature feat = null;
Geometry geom = null;
DoubleRect featBounds = new DoubleRect();

// create a new queryparams that returns geometry only
QueryParams qp = new QueryParams(
    SearchType.entire, true, false, false, false, false);
FeatureSet fs = null;
try {

    // return geometries only
    fs = lyr.searchAll((List)null, qp);

    // initialize layerBounds
    layerBounds.initBounds();

    // get next feature in fs
    feat = fs.getNextFeature();
    while ( feat != null ) {

        // get the bounds of each geometry in the FeatureSet
        geom = feat.getGeometry();
        if ( geom != null ) {

            // build the bounds of the layer by merging the
            // bounds of the feature... with the total
            // rectangle
            layerBounds.merge(geom.getBounds());
        }

// get next feature in fs
        feat = fs.getNextFeature();
    }
}
catch (Exception ex) {
    ex.printStackTrace();
}
```

```
finally {
    try {
        if ( fs != null ) {
            fs.dispose();
        }
    }
    catch (Exception ex1) {}
}

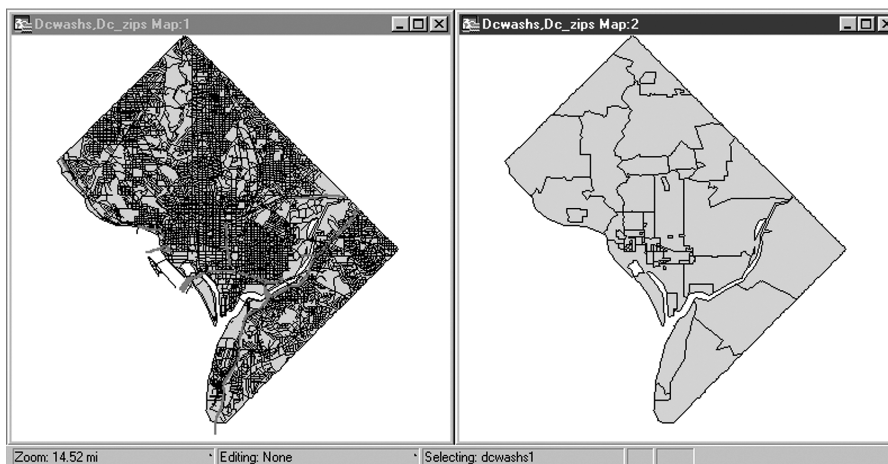
// add a bit of padding
if ( layerBounds != null && layerBounds.area() != 0.0 &&
layerBounds.area() != Double.POSITIVE_INFINITY )
{
    layerBounds.set(layerBounds.center(), layerBounds.width() * 1.05,
            layerBounds.height() * 1.05);
}
```

缩放图层

有时可能只需在特定的缩放级别显示图层。缩放图层控制图层的显示，以便只在地图的缩放级别处于预设距离之内时显示。各个图层可设置不同的缩放图层级别。

例如，如果地图包括街道视图，则可能会发现在用户将地图缩至太小时变得过于模糊。使用缩放图层设置地图，以便 MapXtreme 在用户缩小至特定距离不显示街道，例如 5 英里的距离。

注： 缩放图层是提高渲染地图性能的最重要的因素之一。



以下示例代码通过修改图层对象属性设置缩放图层，以便该图层只在图层缩放介于 10 和 30 km 之间时显示。

```
// set layer for zoom layering from 10 to 30 kilometers
layer.setZoomLayer(true);
layer.setMinZoom(new Distance(10.0, LinearUnit.kilometer));
layer.setMaxZoom(new Distance(30.0, LinearUnit.kilometer));
```

此外还以为各个图层设置不同的缩放级别。例如，有街道图层、县界图层和州界图层。我们希望只有在缩放级别小于 8 英里时令街道图层可见。在缩放级别处于 20 英里和 200 英里之间时显示县界图层。并且希望只有在缩放级别大于 100 英里的时候令州界图层可见。

生成图层标注

如果没有说明图元的标注，地图就无法体现其实际的用途。 MapXtreme 支持多种标注属性，赋予地图易于区别的外观并提供了帮助信息。

直线标注是根据地理对象的标记锚点位置来绘制的。该标注位置近似于对象的中点，但是不一定非是对象中点。曲线标注贴合多义线或多边形路径的曲线。



标注是动态连接到其地图对象的。如果数据或地理信息改变，则标注也会随之改变。标注内容由与地理对象关联的数据确定。

图层可设置为使用 FeatureLayer.**setAutoLabel** 方法自动标注。如果图层为自动标注，则 **isAutoLabel** 将返回 True 或 False。

除了标注内容之外，还可以通过使用 LabelProperties 类的方法来控制自动标注的显示和样式。也可设置用于显示标注的条件、显示的样式，以及其对于图层中所有对象的优先级。

标注是 MapXtreme 的强大功能之一，我们为此专门对其进行了介绍。请参阅*第 14 章：标注和样式* 。

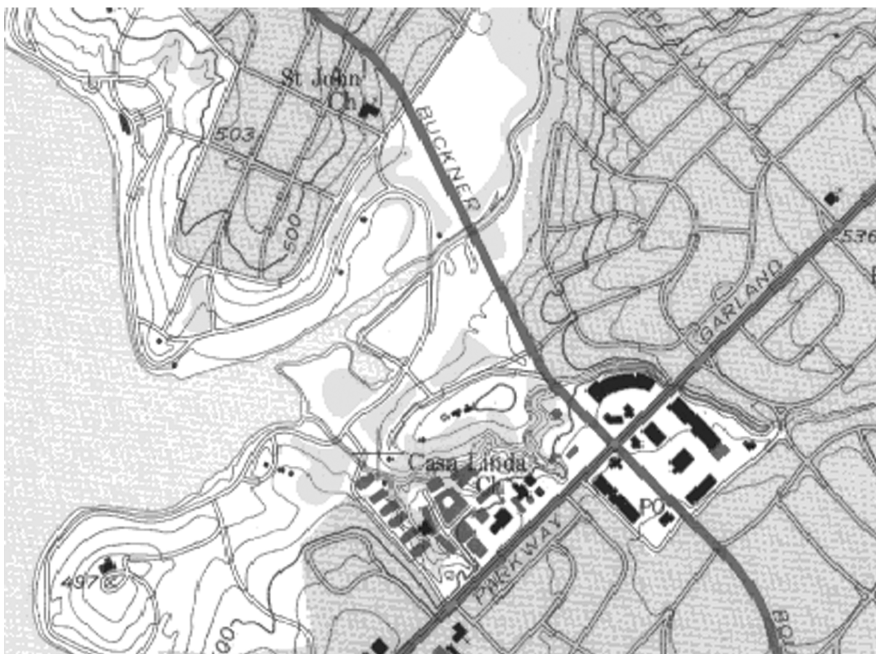
栅格图像

栅格图像是可以包含在地图中的另一种图层。栅格是计算机化的图片，其中包含一行一行的小点 （像素）。有时也称其为位图。航拍照片和卫星成像是 GIS 中最常见的栅格数据。

在 MapXtreme Java 应用程序中，可以将栅格图像显示为所创建的地图的背景幕。随后，即可在图像上覆盖附加数据，例如街道地图和客户位置。

注： 本节介绍将栅格图像引入 MapXtreme Java 的有关信息。不要将这一点和完整地图的栅格图像输出混为一谈，该输出由 MapXtremeServlet 在 HTML 页面中返回。有关栅格输出的详细信息，请参阅*第 11 章：渲染涉及的考虑因素* 。

要将栅格图像显示为地图图层，该图像必须包含地理注册信息，即相应于地理位置的坐标。以便定义地图图像的正确位置。



MapXtreme Java 支持 2 种类型的栅格图像：

- 使用包含地理注册信息的关联 .tab 文件的图像。此类栅格图像包括 TIFF、 JPEG、 GIF、 BMP、 PNG、 XBM 和 MIG （MapInfo 网格）。
- 注册信息包含在图像文件特殊标记中的图像。此类格式包括 GeoTIFF 和 MIG[1]。

要将图像注册为地理信息正确的图像，可以将图像引入 MapInfo Professional 并进行注册。许多 USGS 地图图像都带有关联的 .tab 文件。

将栅格图层添加到 MapJ

栅格图像引入地图的方式和添加其他图层的方式相同 — 通过创建说明图像以及其位置的数据提供方。如果栅格带有关联的 .tab 文件，则需要创建 TABDataProvider。对于 GeoTIFF 图像，则需要创建 GeoTIFFDataProvider。

MapXtreme Java 实施灵活的栅格处理方案，可用于动态创建数据提供方。如果栅格注册到 TAB 文件，则 TAB 数据提供方将读取 RasterDataProviderFactory.xml，确定在加载栅格文件时所要使用的栅格数据提供方。 RasterDataProviderFactory.xml 是由栅格数据提供方来组织的。每个栅格数据提供方条目都有一个文件后缀列表 （如 gif、 jpeg 等），表示特定栅格数据提供方可以读取的栅格格式类型。查看随 MapXtreme Java 提供的 RasterDataProviderFactory.xml，即可发现用于读取 GIF、 JPG、 PNG 和 JPEG2000 栅格文件的是调用 com.mapinfo.dp.jai.JaiDataProvider 的栅格数据提供方。 JaiDataProvider 是随 MapXtreme Java 提供的数据提供方中的唯一栅格数据提供方。此类使用 Java 高级图像 (JAI) API 用于加载栅格文件。与 RasterDataProviderFactory.xml 中的 JaiDataProvider 关联的后缀列表并非完全列表，用户可以通过参数来读取 JAI 可以读取的任意栅格格式。

注： 如果图像无法由以上栅格数据提供方处理，则将会报错，表示无法处理的特定文件。

GeoTIFF 文件可以添加到 MapJ，正如使用 GeoTIFF 数据提供方的任何其他图层一样。

以下代码说明了如何创建 GeoTIFFDataProvider 以及将 GeoTIFF 图像添加到地图。此时，图像将存储在本地系统之中，并将由 LocalDataProviderRef 进行检索。

```
// Create a TableDescHelper that points to the Tiff image
GeoTIFFTableDescHelper geoTiffTDHelper = new
GeoTIFFTableDescHelper("e:\\image\\dcquad.tif");

// Create DataProviderHelper
// (**note this constructor takes no parameters)
GeoTIFFDataProviderHelper geoTiffDPHelper = new
GeoTIFFDataProviderHelper();
```

1. 对于 MIG 文件而言，尽管没有必要具有关联的 .tab 文件，但是不能直接使用 MapXtreme Java 管理器直接打开 MI Grid 图像。取而代之的是，打开关联的 tab 文件。

```
// If the data is local, use a LocalDataProviderRef
LocalDataProviderRef localDPRef= new
LocalDataProviderRef(geoTiffDPHelper);

// Insert the layer into the map layer collection
map.getLayers().addLayer(
    localDPRef, geoTiffTDHelper, "GeoTIFF Layer");
```

导入栅格图像的考虑因素

以下内容介绍导入栅格图像时的注意事项。

设置栅格坐标系的显示

在向地图添加栅格图像之后，确保将 MapJ 的显示坐标系设置为栅格图层的坐标系，其原因在于 MapXtreme Java 不会对栅格图像进行再次投影。

以下代码示例显示了如何确定用于栅格图层的坐标系，以及设置相应坐标系的方法。

```
TableInfo ti= rasterLayer.getTableInfo();
CoordSys cSys= ti.getCoordSys();
myMapJ.setDisplayCoordSys(cSys);
```

栅格和性能

由于在显示和使用栅格图像时增加了对系统的要求，因此我们强烈建议使用 64 MB 或更大内存来扩展堆大小的上限 （具体取决于所用应用程序和栅格文件类型）来启动服务器或应用程序。

例如，要在使用 MapXtreme Java 管理器加载图层时增加堆大小的上限，可在命令行键入：

```
java -mx64M com.mapinfo.mjm.client.MJMClient <MJM Servlet URL>
```

本地渲染栅格图像

所有栅格图像的任意处理都必须在本地完成。 Renderer 对象和图像文件必须位于同一计算机上，因为两者都需要作为随机访问文件读取。

图像 IO 数据提供方

MapXtreme Java 包括一个数据提供方，允许第三方以其地图绘制应用程序来显示定制栅格图像格式。这是通过 Sun 的 ImageIO 接口来支持的，该接口包含在 Java 2 Platform SDK 1.4 之中。这些定制格式可以通过 IIODataProvider 读取到 MapXtreme Java 中，其前提是有一个包含地理注册信息的关联 .tab 文件。 IIODataProvider 包含在 com.mapinfo.dp.imageio 之中。

要使用 MapXtreme Java ImageIODataProvider，必须先创建插件，此时务必遵循 Sun 的网站上提供的 Sun 图像 IO 指南。随后必须修改 rasterhandlerfactory.properties 文件，如下所示。计划在可用的栅格数据提供方的列表中提供 com.mapinfo.dp.imageio.IIODataProvider。务必在 JDKRasterDataProvider 之前添加 IIODataProvider，以便在读取相同格式时使用 IIO。有关 rasterhandlerfactory.properties 的详细信息，请参阅第 191 页的*将栅格图层添加到 MapJ* 。

栅格样式标记

栅格样式标记包含在保存栅格注册信息的同一 .tab 文件之内。这些标记说明图像的特定显示效果，其中包括亮度、对比度、灰度、透明度、半透明度和网格。

注： MapXtreme Java 只能读取这些样式标记。要对样式作出任何更改，必须在 MapInfo Professional 打开栅格图像，并在 “调整图像样式”对话框中更改样式。保存这些图像时， .TAB 文件将使用新的样式信息更新。

栅格样式可以影响到渲染进程的速度。在 MapXtreme Java 中，样式是通过向渲染链添加过滤器来进行渲染的。透明和半透明效果是耗时最多的操作。

务必不要将这些栅格图像和 MapXtreme Java 渲染的输出栅格图像相混淆。输出栅格图像由图元所在的图层构成，相应的图元有可能包含注册的栅格图像，也有可能不含注册的栅格图像。

以下是栅格图像注册文件的示例。栅格样式标记显示为粗体。下表说明了标记的编号和值。

```
!table
!version 300
!charset WindowsLatin1
Definition Table
  File "conus13.tif"
  Type "RASTER"
   ...
  RasterStyle 1 62
  RasterStyle 2 40
  RasterStyle 3 1
```

```
RasterStyle 4 1
RasterStyle 7 1525779
RasterStyle 8 221
```

下表说明了有效的栅格样式标记编号和相应的值。如果在 .TAB 文件中没有出现用于该样式的标记，则将使用默认值。

栅格样式	说明	标记编号	值
亮度	影响图像显示的明暗。	1	0-100，默认为 50。
对比度	影响阴影区域之间差异的大小。	2	0-100，默认为 50。
灰度	切换：确定图像显示为黑白影线还是彩色表示。	3	0 = off, 1 = on，默认 0。
透明度	切换：确定颜色是否渲染为透明。	4	0 = off, 1 = on，默认 0。
网格	确定栅格是否为网格。	6 *	1 = yes, 0 = no. 默认为 0。
透明度 BGR	确定将哪一颜色渲染为透明。	7	BBGGRR 是分别用于蓝色、绿色和红色的颜色值，表示要渲染为透明的颜色。无默认值。
半透明度	确定显示图像的透明度。	8	0-255，由图像的 alpha 通道值确定。 0 = 100 半透明百分比 （不可见）， 255 = 0 半透明百分比 （不透明）。无默认值。

* 标记 5 已经逐渐停用。

亮度和对比度

亮度和对比度影响栅格图像的整体显示。更改亮度和 / 或对比度可以有助于更好地区分图像图元。亮度影响图像显示的明暗。亮度值越高，图像显示就越亮。对比度影响区域之间的阴影。较高的值表示图像的特定区域比其他区域更加突出。

亮度和对比度值介于 0 至 100% 之间。如果该标记没有出现，则默认值为 50。这意味着对栅格图像不应用任何对比度或亮度。

灰度

以黑白阴影显示的图像称为灰度图像。如果栅格图像 .TAB 文件包括栅格标记 3 1，MapXtreme Java 将彩色图像显示为灰度。如果要将该图像打印到灰度打印机，则这一特性非常实用。

透明度

MapXtreme Java 支持单色透明度，此时栅格图像的一种颜色在图像显示时可成为透明或不可见。如果要在其他图层上显示图元，而这些图元在正常情况下为图像中的相应颜色所遮盖，那么这一特性将会将会非常实用。

透明度通过两个样式标记在 .TAB 文件中表示：标记 4 1 表示该透明度打开，标记 7 加 BBGGRR 形式的数字表示要显示为透明的颜色。 BBGGRR 是表示为整数的十六进制值，用于表示构成颜色的蓝、绿和红色值。例如，用于 RasterStyle 标记 7 的 1525779 表示暗绿色的影线表示。

注： 数字 1525779 是十六进制值 174813 的十进制表示形式，分别表示蓝色 23(0x17)、绿色 72(0x48) 和红色 19(0x13)。

左下方图像表示地图中间从上至下的高速公路，右图所示是在 MapInfo Professional 中显示为透明之后的效果。



网格

网格是一个采用连续渐变颜色的图像，通常可用于表示海拔或温度的变化。 MapXtreme Java 可以确定在关联 .TAB 文件中提供栅格样式标记 6 1 的情况下，注册栅格图像是否为网格。如果该标记没有出现，则图像将按照常规处理。

半透明度

半透明度是图像的透光量，图像将显示为半透明或透明。它由 RasterStyle 标记 8 后跟 0 至 255 之间的数字来表示。该数字表示作为图像组成一部分的 alpha 通道值，说明其透明的程度。图像的 alpha 通道值越低，图像显示的半透明效果就越强。 alpha 数值为 0 时，将图像渲染为完全不可见 （100% 半透明）。采用 alpha 值 255，图像将完全不透明 （0% 半透明）。

对于 MapXtreme Java 中支持的所有栅格样式，半透明对于图像渲染速度的影响最大。这是因为该操作影响图像的每个像素，无论该像素是什么颜色。对于 Java 而言， 0 至 255 之间的值的渲染本身就较慢。

MapXtreme Java 中的栅格图像

网格图像是显示表示特定值的连续渐变颜色的特殊栅格图像类型。网格地图通过在网格单元集合中插补取自源表的点数据创建。

网格图像和常规栅格图像有所不同的是， MapXtreme Java 可以返回存储在单元中的值，无论相应的值是连续网格中的插补值，还是与分类网格中的单元关联的名称。使用 FeatureLayer.searchAtPoint(...) 或 InfoTool 来检索该信息。

支持两种类型的网格图像：

- MI 网格
- Vertical Mapper 网格

MI 网格栅格

MI 网格栅格文件 (.MIG) 是通过 MapInfo Professional 创建的专题影线表示的网格地图。MIG 文件既可通过使用 TAB 数据提供方加载引用 MIG 文件的 TAB 文件添加到 MapJ，也可直接使用 MIGrid 数据提供方来添加到 MapJ。MapXtreme Java 示例数据中已包含用于美国降雨量、温度和海拔的网格文件。有关创建 MI 网格文件的详细信息，请参阅 MapInfo Professional 文档。

Vertical Mapper 网格支持

MapXtreme Java 可读取和显示 Vertical Mapper 网格文件。这些文件在 MapInfo Professional 中使用 Marconi 的 Vertical Mapper 插件来进行创建和修改。

要在 MapXtreme Java 中显示 Vertical Mapper 网格文件，可使用 TAB 数据提供方或 Northwood Grid 数据提供方。没有 Vertical Mapper 网格的 .TAB 文件，就无法进行加载。务必确保 NwSGridReader.jar 和 rfgrid.jar 均位于系统 classpath 之中。ABDataProvider 将该文件识别为 Vertical Mapper 网格，并调用 NWGridDataProvider。

Vertical Mapper 网格可以采用以下两种类型之一：连续 (.grc) 或分类 (.grd)。连续的网格在表示单元之间变化的地图上显示为颜色渐变，例如在海拔地图上。连续的网格可用于估计实际的数据集合点之间的位置值。

分类网格地图包含单元，单元地图中的值直接成为表示图元的类的颜色。例如，在显示树的种类分布的造林网格地图上，可以采用蓝色表示蓝云杉树，用红色表示橡树，用黄色表示桦树。

11 渲染涉及的考虑因素

渲染特指生成地图图像的过程。渲染图像是创建基本地图的最后一个步骤。本章介绍各种渲染选项和考虑因素。

本章内容：

MapXtremeImageRenderer

远程渲染使用 MapXtremeImageRenderer 来处理。

远程渲染表示包含 MapJ 对象的应用程序将服从 MapXtremeServlet 的实例来创建地图。MapJ 客户机将其请求 （如何渲染和渲染什么）发送到 MapXtremeServlet，由后者处理请求，然后将结果返回给 MapXtremeImageRenderer，可能的三种结果如下所示：

- 本地系统上使用 **toFile()** 的位图文件 （GIF、 JPEG 等）－ 通常用于在中间层上存储图像，然后令浏览器从服务器请求文件。
- 使用 **toStream()** 的 Java 输出流对象 － 流可以表示为文件 （与以上相同）或者内存存储的信息 （作为 Java 图像对象，如下）；两层结构使用 toStream，客户机可将图像存储在内存中。
- 使用 **toImage()** 的 Java 图像对象 － 将栅格保存在内存中，可直接用于显示。

由于要赖于客户机系统可能没有的 JVM 和字体等系统资源，因此 MapXtremeImageRenderer 是渲染地图最普通的方式。

LocalRenderer

LocalRenderer 将地图渲染为 Java2D Graphics2D 或 BufferedImage 对象。 Graphics2D 对象通常源自 BufferedImage 或 Swing 组件。所有的渲染均在客户端计算机完成。由于所有渲染均在客户端计算机完成，渲染地图所需资源必须驻留在客户端计算机 （字体、视频卡等）。默认情况下， VisualMapJ 将使用 LocalRenderer。

EncodedImageRenderer

MapXtreme Java 提供特殊渲染器来帮助用户使用动画图像创建地图。有关详细信息，请参阅第 202 页的*动画图像* 。

使用命名地图渲染附加图层

命名地图最初于 MapXtreme Java 4.0 中引进。借助于命名地图，可采用更加便于日后记忆的名称来保存图层集合。相应操作可通过 MapXtreme Java 管理器的保存地图定义 （请参阅*第 5 章：管理 MapXtreme Java* ）或使用 NamedMapDefContainer （请参阅*第 9 章：MapJ API* ）编程来完成。

改进的编程接口可用于再次调用命名地图，并包括一个在命名地图顶部渲染的图层列表。如果要在注释图层 （Annotation 图层）中突出显示特定图元，例如两点之间的路线，那么这一特性尤为实用。与此同时，这还是请求地图最为经济的方法，因为其向服务器发送的渲染请求 “较瘦”。此前的进程需要在客户机渲染命名地图，然后将图层列表添加到地图，再将所有要渲染的图层发送到服务器。

通过编程，客户机使用 ImageRequestComposer 的新工厂方法，可创建 Enterprise XML 的地图图像请求，其中包括命名地图的名称、 MapJ 和 overlayIndex，后者是一个源自客户机需要在命名地图上渲染的 MapJ 对象的图层列表。图层可以采用可支持类型 （注释、 TAB、数据库、查询构建器或数据绑定图层）的任意组合。

代码示例

在本例中，已经创建基础地图并将其另存为命名资源。此外还创建了另一个包含多个图层的地图并将其另存为地图定义文件。 MDF 将加载到 MapJ 对象之中。在创建 ImageRequestComposer 对象之后，命名资源将被构建器中传递的整数值所指定编号的图层所覆盖。

```
// Set mime type
private static String mimeType  = "image/gif";

// Specify a map
private static String mymapPath =
    "C:\\CODE_SAMPLES\\NAMEDMAPS\\POINTS.mdf";
private static String mapName   = "RENSS_CTY";
private static String providerURL =
    "file:///C:/cat3345/webapps/mapxtreme47/resources/codesample";
private static String mymxtURL =
    "http://stockholm:8080/mapxtreme47/mapxtreme";
private static final int ovrlayIndex = 1;
public void service(HttpServletRequest request,
    HttpServletResponse response)throws IOException, ServletException
    {
    response.setContentType("image/gif");
    ServletOutputStream sos = response.getOutputStream();
    MapJ myMap = new MapJ();
        myMap.loadMapDefinition(mymapPath);
```

```
        myMap.setDeviceBounds(new DoubleRect(0,0,800,600));
        try
        {
            //myMap.setZoom(zoom);
            ImageRequestComposer irc =
                ImageRequestComposer.create(providerURL,mapName,myMap,
                ovrlayIndex,ImageRequestComposer.MAX_COLORS_TRUECOLOR,
                Color.white,mimeType);
            MapXtremeImageRenderer renderer = new
                MapXtremeImageRenderer(mymxtURL);
            renderer.render(irc);
            renderer.toStream(sos);
            renderer.dispose();
        }
        catch (Exception e)
        {
            System.out.println("Error");
            e.printStackTrace();
        }

        }
    }
```

动画图像

MapXtreme Java 提供了使用动画图像构建地图所需的类。相应功能只适用于点图元。

按照 MapXtreme Java 术语，动画图像称为覆盖图像。覆盖图像是 Rendition 对象的属性。当 Rendition.SYMBOL_MODE 属性设置为 Rendition.SymbolMode.**OVERLAY_IMAGE** 时，Rendition.SYMBOL_URL 属性将用于从指定 URL 检索图像。（对于生成任意类型的图像符号而言均为同样的行为。）

为表明所需的动画图像信息， Enterprise XML 协议中的 MapImageRequest 已经改进，以返回图像 （image/gif、 image/jpeg 等）和 **MapImageResponse**。响应为包含基础地图的 XML 文档和点覆盖的列表。每个点覆盖元素均包含说明其样式和相对于基础地图位置的有关信息。

MapImageResponse 将通过新的渲染器 **EncodedImageRenderer** 生成。 MapImageResponse 采用名为 application/encodedimage+xml;image/xxx 的新 MIME 类型，其中 xxx 可以是 gif、jpg、 png 等类型。

使用 MapJ API 的操作步骤如下：

1. 将点图元的样式设置为动画（样式 SYMBOL_MODE 属性设置为 Rendition.SymbolMode.OVERLAY_IMAGE）。
2. 指定渲染响应是 application/encodedimage+xml;image/xxx, MIME 类型为 application/encodedimage+xml。
3. 使用 EncodedImageRenderer 渲染。
4. 检索包含基础静态地图和覆盖图层列表信息的 MapImageResponse 文档。

具备这一信息，即可准备就绪，开始创建地图。

由于 MapJ API 符合公共的 Enterprise XML 协议，因此还可以完全在 MapJ API 之外生成这一信息。此时的操作步骤与此前比较类似，如下所示：

1. 将点图元的样式设置为动画（样式 SYMBOL_MODE 属性设置为 Rendition.SymbolMode.OVERLAY_IMAGE）。
2. 指定渲染响应为 application/encodedimage+xml。
3. 向 MapXtremeServlet 发送请求。
4. 对 XML 响应进行语法分析。

动画图像的使用情况

为了进一步说明如何使用动画图像，可以考虑以下情况。假定要使用动画 GIF 在 Web 浏览器中显示飘动的旗帜作为点图元，用于标记全球各地办公室的位置。所创建的 JSP/Servlet 将使用此前通过上述过程所接收到的 MapImageResponse 信息来渲染地图。静态基础地图以及其动画图像置于要在此后访问的 “图像服务器”中。随后 JSP/Servlet 将创建一个带有图层的 DHTML 页面。最底层图层将具有静态基础地图，该地图包含指向图像服务器的 URL。第二个图层将具有包含访问图像服务器的 URL 动画图像。浏览器中的效果是该图像显示在基础地图之上。

以下是说明相应图层的 DHTML 页面示例。

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
  <title>Animated Icons Test</title>
</head>

<body>
  <div ID="wa_flag" STYLE="position: absolute; left: 75; top: 20;">
    <img SRC="http://stockholm:8080/mapxtreme/servlet/
    imageserver?name=washington_md_wht_8760.gif" >
  </div>
  <div ID="ca_flag" STYLE="position: absolute; left: 75; top: 150;">
    <img SRC="http://stockholm:8080/mapxtreme/servlet/
    imageserver?name=california_full_md_wht_7881.gif" >
  </div>
```

```
    <div ID="ny_flag" STYLE="position: absolute; left: 500; top: 75;">
      <img SRC="http://stockholm:8080/mapxtreme/servlet/
      imageserver?name=new_york_full_md_wht_7193.gif" >
    </div>
      <img SRC="http://stockholm:8080/mapxtremeserver/servlet/
      imageserver?map=some_unique_id" height=361 width=589 align=LEFT>
  </body>
  </html>
```

这一 DHTML 文件在中间层创建，用于通过 Web 浏览器显示。此应用程序具有若干个已经注册到用户创建的 “图像服务器” 的标准动画图像。在通过中间层渲染之后，生成的静态地图已经注册到图像服务器，可在此后由客户机的 Web 浏览器下载。由于静态地图图像对于客户机应该是唯一的，因此其应该采用唯一的名称。 DHTML 页面将返回到客户机的浏览器，此后浏览器将尝试下载在页面的 <IMG> 标记中引用的图像。此时由于 <DIV> 标记的存在，可产生动画效果，该标记允许 HTML 图层覆盖基础 HTML。

代码示例

在 examples/server/java 目录中提供的是一个示例应用程序，展示覆盖图像如何显示在瘦客户机方案的地图上。在本例中，动画图像是突出显示各国首都的星型符号。要运行该 servlet，可打开位于 webapps/samples47 中的 index.htm，然后单击指向覆盖图像示例 Servlet 的链接。

栅格输出格式

MapXtreme Java 支持多种栅格输出格式，其中包括 GIF、 JPEG、 PNG 和 WBMP。栅格图像的输出格式在 ImageRequestComposer 中通过 MIME 类型指定。MIME 是一种用于图像数据等非文本数据的格式标准。以下指南将帮助您确定最适合具体需求的格式类型。

- image/jpeg — JPEG — 适用于颜色多于 256 种的图层。
- image/gif — GIF — 适用于最多 256 色的图层。
- image/png — PNG — 是 GIF 格式的替代格式，适用于多于 256 色的图层。
- image/wbmp — WBMP — 用于在类似手机和 PDA 的手持设备中生成图形的专用格式。
- image/svg+xml — SVG — 用于在 XML 中说明二维向量图形的格式。

例如，要输出 JPEG，可使用如下所示的 ImageRequesComposer：

```
ImageRequestComposer.create(mapj, maxColors, bgColor, "image/jpeg");
```

在使用栅格文件时，建议使用 PNG 输出。 GIF 输出限定为最高 256 色和，但栅格文件通常至少具有 256 RGB 或灰度颜色。添加向量图层可能会引入 256 种以上的颜色，此时必须减少颜色种类，这是一个相当耗时的操作。

设置 JPEG 图像的质量

要控制 JPEG 输出在服务器上的质量，需要在 servlet 容器中设置 **jpegQuality** 参数。例如在 Tomcat 中，可编辑 /mapxtreme47/WEB-INF 目录下的 web.xml 文件，纳入一个用于 JPEG 质量的值。

```
<init-param>
    <param-name>
        jpegQuality
    </param-name>
    <param-value>
        85
    </param-value>
</init-param>
```

jpegQuality 值介于 0ñ100 的范围之间，默认值为 75。图像的质量随该值的减小而降低，此时将生成较小的图像。

SVG 输出

MapXtreme Java 版支持采用 SVG 格式导出地图图像，此图形格式用于在 XML 中描述二维图形。要将导出格式指定为 SVG，可在 ImageRequestComposer 中设置 MIME 类型。例如：

```
ImageRequestComposer.create(
    mapj, maxColors, bgColor, "image/svg+xml");
```

MapXtreme Java 版中的 SVG 兼容 SVG 1.0。SVG 为在小型设备上渲染地图提供了便利，并且提供了一个独立于分辨率的渲染格式。

使用 Rendition.SYMBOL_URL 属性，可将栅格数据直接嵌入到 SVG 文档中。有关 MapXtreme Java 版中的嵌入式栅格数据的详细信息，请参阅第 266 页的*图像符号*。有关数据 URL 方案的详细信息，请访问 **http://www.ietf.org/rfc/rfc2397.txt**。

向 SVG 添加 JavaScript 事件

SVG 文档还具有一个与 HTML 文档中的事件处理程序类似的事件处理程序工具。 SVG 文档可以提供若干种事件类型的信号，例如鼠标移动、重新调整大小等。这一特性可用于实现交互操作，例如通过浏览器使用 JavaScript 实现交互。为使用 SVG 事件， SVG 文档应该

嵌入到其他文档中，例如嵌入到提供实际处理程序实现的 HTML 文档中。 MapXtreme Java 允许用户通过 com.mapinfo.xmlprot.mxtj.SvgConditions 类指定用于所有 SVG 事件的事件处理程序。以下示例展示了如何为 SVG onmouseover 事件指定具体的功能：

```
// create a Map that associates a function with an SVG event
Map eventMap = new HashMap();
eventMap.put("onmouseover", "OnMouseOver_Event");

// create an SvgConditions object
SvgConditions svgConditions = new SvgConditions(eventMap);

// create an ImageRequestComposer that signifies we want SVG as our
// render format
ImageRequestComposer irc = ImageRequestComposer.create(
   mapj, 257, Color.RED, "image/svg+xml");

// set the SvgConditions
irc.setSvgConditions(svgConditions);

// render
renderer.render(irc);
```

有关如何控制 SVG 事件功能的更多示例，请参阅 MapXtreme Java 的 SVG JSP 地图查看器。

MapXtreme Java 的 SVG 文档兼容 SVG v1.0。

MapXtreme Java 中 SVG 的限制

在使用 SVG 和 MapXtreme Java 时，具有以下限制：

- SVG 图像不能在本地渲染。
- MapXtremeImageRenderer 的方法 toImage() 不能与此版本的 MapXtreme Java 中的 SVG 一起使用。
- 渐进渲染不能和此版本的 MapXtreme Java 中的 SVG 一起使用。

WBMP 支持

MapXtreme Java 版支持采用 WBMP 格式导出地图图像，此图形格式用于类似手机和 PDA 的手持设备。要将导出格式指定为 WBMP，可在 ImageRequestComposer 中设置 MIME 类型。例如：

```
ImageRequestComposer.create(mapj, maxColors, bgColor, "image/wbmp");
```

MapXtreme Java 中的 WBMP 支持包括两种输出模式：快速地图显示阈或抖动显示。后者效果好但速度慢。各种类型的说明如下。

WBMP 输出的阈方法

阈方法是一项基本技术，采用该技术时，地图中每个颜色的像素都将转换为灰度，然后与设置阈比较，生成 0 （黑）或 1 （白）的表示。阈值可在 servlet 初始化例程中配置。

该算法用于确定是否将特定像素转换为 0 或 1，如下所示：颜色转换为灰度 (R+B+G/3)，其中 R、 B 和 G 分别表示像素的红、绿、蓝三色的色值 (0, 255)。如果值大于阈值，则像素设置为 1 （白）。如果值小于或等于阈值，则像素颜色为 0 （黑）。默认阈值为 127。

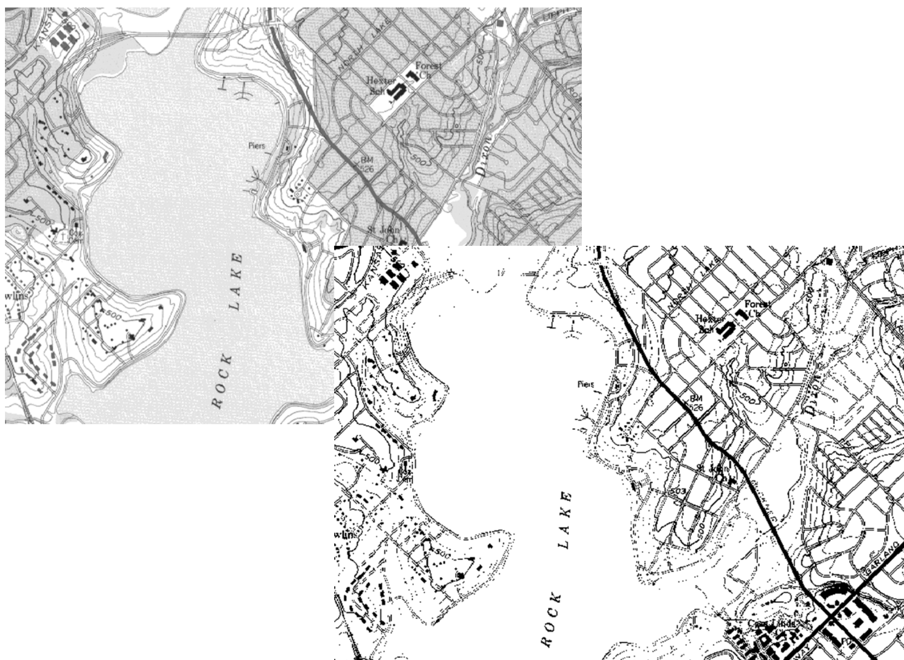
默认情况下，如果将 MapXtremeImageRenderer MIME 类型指定为 image/wbmp，则 MapXtreme Java 将自动执行地图阈的转换。

要更改默认阈值 127 以生成其他图像质量，必须在 servlet 容器的初始化例程中指定新值。

例如，以下是使用 Tomcat 时所添加的代码。

```
<servlet>
  <servlet-name>mapxtreme</servlet-name>
  <servlet-class>com.mapinfo.mapxtreme.MapXtremeServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param>
    <param-name>wbmpThreshold</param-name>
    <param-value>95</param-value>
  </init-param>
</servlet>
```

以下图解近似显示了原色 GIF 图像 （左）和使用阈方法导出为黑白 WBMP 的相同图像。



WBMP 输出的抖动方法

与阈方法相比，抖动地图图像将生成较高质量的地图显示，但是其绘图速度较慢。

抖动考虑了像素颜色的转换因素，创建了一种应用于使用该颜色的区域的抖动模式。抖动是一个调整不同颜色相邻元素，为位于浏览器当前调色板颜色之外的颜色提供虚拟色的过程。 MapXtreme Java 中当前使用的抖动例程是 Floyd-Steinberg 错误抖动例程。此方法将像素量化 （颜色减少）期间的错误分布在相邻的像素之间。这样增加了图像透明色的分辨率，这是纯粹抖动无法实现的效果。

要设置图像导出为抖动格式，必须在 servlet 的初始化例程中设置抖动参数。例如，用于 Tomcat servlet 的代码示例如下：

```
<servlet>
  <servlet-name>mapxtreme</servlet-name>
  <servlet-class>com.mapinfo.mapxtreme.MapXtremeServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param>
    <param-name>wbmpDither</param-name>
    <param-value>1</param-value>
  </init-param>
</servlet>
```

以下图解近似显示了原始在线彩色 GIF 图像 （左）和使用抖动方法导出为黑白 WBMP 的相同图像。



复合渲染器

MapXtreme Java 的渲染器称为复合渲染器，可用于指定在图形更新时需要重画的图层。这在只有部分图层包含更改信息的时候尤为实用。 CompositeRenderer 可用于将 MapJ 中的图层划分为静态和动态渲染的图层。静态图层将只能重画一次，并在本地存储为位图。动态图层将根据每个渲染请求重画。

复合渲染器提供实用的应用程序，例如实现了在地图上显示地理编码的点或是显示沿公路移动的车辆等功能。

要在地图上显示地理编码的点，可在基础地图上创建一个注释图层 （Annotation 图层），保存表示地理编码位置的图钉符号。采用 CompositeRenderer，可渲染带有标注的基础地图图层，图钉将出现在这些标注之上。如果不采用 CompositeRenderer，标注将始终最后绘制，即存在令其下面的信息变得模糊的可能。

在地图上显示移动的对象。 GPS 应用程序可以显示一辆沿地图上的高速公路移动的卡车。基础地图只需绘制一次，而卡车符号将在需要时随时绘制 （如每个位置改变）。

使用 CompositeRenderer 时切记以下信息：

- 静态图层将绘制为本地存储的位图。此时将使用大量内存，一个 640 x 480 的图像将至少占用 2.7MB 内存。
- 更改 MapJ 上的缩放将导致静态图层变形。直线将显示为阶梯状或锯齿状。
- 更改 MapJ 的中心将导致静态图层的边缘平移。此时，可以重新生成静态图层。

代码示例 :Animation 图层

本示例显示如何在地图上创建移动的对象。有关代码示例也可参阅 CompositeRenderer 下的联机 Javadoc。

```
try {
/* ASSUMPTIONS:
  The variable mapj is of type MapJ and has loaded a map
  The variable req is of type HttpServletRequest
  The variable res is of type HttpServletResponse
  The variable dp is of type DoublePoint
*/
// Add annotation layer - this layer will consist of one image symbol
// to "animate"
AnnotationTableDescHelper atdh = new
AnnotationTableDescHelper("Animation_Layer");
AnnotationDataProviderHelper adph = new
AnnotationDataProviderHelper();
LocalDataProviderRef ldpr = new LocalDataProviderRef(adph);

// Add the annotation layer
Layer animate_layer = mapj.getLayers().insertLayers(
    ldpr, atdh, 0, "Animation_Layer");

// Create the rendition for the point
Rendition r = RenditionImpl.getDefaultRendition();
r.setValue(Rendition.SYMBOL_MODE, Rendition.SymbolMode.IMAGE);
r.setValue(Rendition.SYMBOL_URL, "file:///C:/images/car.gif");

//nCreate the label rendition for the point
Rendition lr = RenditionImpl.getDefaultRendition();

//Create the point
FeatureFactory ff = mapj.getFeatureFactory();
// An array containing a single int Attribute
Attribute[] aAIntAttribute = {new Attribute(33)};

//create a new Primary Key an an integer
PrimaryKey pkey = new PrimaryKey(aAIntAttribute);
Feature f = ff.createPoint(dp, r, lr, aAIntAttribute, pkey);
PrimaryKey pk = animate_layer.addFeature(f);
```

```
// Create the ImageRequestComposer
ImageRequestComposer imageRC = ImageRequestComposer.create(
    mapj, ImageRequestComposer.MAX_COLORS_TRUECOLOR, Color.blue,
    "image/gif");

// Create the composite renderer
// Render the image
// Stream the image back to the client
CompositeRenderer compositeRenderer = new CompositeRenderer(
    "http://localhost:8080/mapxtreme47/mapxtreme", 0);
compositeRenderer.render(imageRC);
javax.servlet.ServletOutputStream sos = res.getOutputStream();
compositeRenderer.toStream(sos);

// Set this attribute to false so that the bottom image is not rendered
// next time
compositeRenderer.setRedrawBottom(false);
} catch(Exception e) {
// Take appropriate error handling steps
}
```

渐进渲染

渐进渲染是 MapXtreme Java 提供的强大功能之一，用于先将部分地图图像发送到客户机，并随之在指定的时间间隔之内，发送更加完全的图像，直至完整图像接收完毕。与在整个地图图像完成渲染之后再发送图像相比，这一特性可以更快地实现接收图像的某些部分。

渐进渲染在需要重新生成花费大量时间访问底层数据的图层时非常实用。那么采用渐进渲染与发送一个最终的图像相比，显示整个图像所用的总时间会略有增加。

客户机控制图像是否采用渐进渲染，以及采用多长的时间间隔来通过 ImageRequestComposer 使用的特殊 MIME 类型返回。这一 MIME 类型采取以下形式：

```
multipart/image;imagetype=xxx;interval=yyy
```

其中 xxx 是要返回的图像的 MIME （如 image/gif、 image/jpg 等）， yyy 是以毫秒为单位的更新时间间隔。

如果启用渐进渲染，那么将由 MapXtremeImageRenderer 上的 toStream()、 toFile() 和 toImage() 方法在流中返回下一图像数据块。布尔值 **isDone()** 方法可用于遍历并检索流中的每个后续图像，直至检索不到为止。以下代码示例说明了 isDone() 的用法。

```
// create a MIME that tells the MapXtremeImageRenderer
// to render the map to PNG every three quarters of a second.
String progressiveMime =
    "multipart/image;imagetype=image/png;interval=750";

// standard ImageRequestComposer using MapJ
ImageRequestComposer irc = ImageRequestComposer.create(
    mapj, MAX_COLORS, Color.WHITE, progressiveMime);

// create MapXtremeImageRenderer
MapXtremeImageRenderer renderer = new
    MapXtremeImageRenderer(MXJ_SERVLET_URL);

// while we are not rendering
while (!isDone()) {
    // get an image from the MapXtreme Servlet
    Image image = renderer.toImage();

    // do something with the image
    ...
}
```

Intra-Servlet 容器渲染器

为了利用在 J2EE 2.2 应用程序中提供的 servlet 转发功能， MapXtreme Java 提供了 **IntraServletContainerRenderer**。这一特性为将栅格图像返回到客户机提供了一种可选途径。这一渲染器在渲染器和 MapXtremeServlet 之间不需要套接字连接，但是这对于 MapXtremeImageRenderer 而言却是不可或缺的。

这一部署选项的优点在于栅格图像可以直接发送到客户机。 MapXtremeServlet 不需要将图像写入到中间层，然后再令中间层将其重写回客户机。但是其限制是应用程序必须部署在和 MapXtremeServlet 相同的容器之中。

IntraServletContainerRenderer 构造器需要输入该信息，以便中间层 servlet 获取 MapXtremeServlet 的 **RequestDispatcher** 对象。RequestDispatcher 对象处理 servlet 转发必需的信息如下所示：

- 由 com.mapinfo.mapxtreme.MapXtremeServlet 使用的别名，例如 mapxtreme47
- MapXtremeServlet 的 ServletContext 对象，或是 servlet 上下文的 URI，例如 /mapxtreme47/servlet
- MapXtremeServlet 将使用 HttpServletRequest 和 HttpServletResponse 对象满足该请求
- 栅格图像的 mime 类型
- 图像是否应该为多个部分，以及多个部分的更新间隔

代码示例 :Servlet 转发

如果 IntraServletContainerRenderer 部署在 MapXtremeServlet 之外的其他 servlet 上下文中，那么在调度请求时，有可能会出现安全问题。

对于 Tomcat 3.x 的用户，要正确使用 IntraServletContainerRenderer，就需要在 conf/server.xml 文件中将以下属性设置为 true，以便用于具有使用 IntraServletContainerRenderer 部署的 servlet 的任意上下文。其示例如下：

```
<Context crossContext="true" ....</Context>
```

不按此操作将导致 MapXtremeServlet 的 ServletContext 在返回时为空。

```
/* Assumptions:
The variable mapj is of type MapJ and has loaded a map.
The variable req is of type HttpServletRequest.
The variable res is of type HttpServletResponse.
*/
// Retrieve the current servlet's ServletConfig object
ServletConfig thisServletConfig = getServletConfig();

// Retrieve the current servlet's ServletContext object from the
//thisServletConfig object
ServletContext thisServletContext =
    thisServletConfig.getServletContext();

/* NOTE: Retrieve the MapXtremeServlet's ServletContext object from
the thisServletContext object. The value of getContext is dependent on
how you deploy the MapXtremeServlet in your servlet container, and is
the URI of the MapXtremeServlet. To access the MapXtremeServlet in this
deployment you would use http://stockholm:8080/mapxtreme47/mapxtreme.
*/
ServletContext mxtServletContext = thisServletContext.getContext(
    "/mapxtreme47/mapxtreme");
```

```
// Check to make sure the mxtServletContext is not null
// (This may be null due to servlet container security)
if(null == mxtServletContext)
{

// Take appropriate error handling steps
}

// Create the IntraServletContainerRenderer
try
{
/* NOTE: The argument of "mapxtreme" on the next line is the name of
the MapXtremeServlet, as defined by the deployment of the servlet
*/
IntraServletContainerRenderer isRenderer = new
    IntraServletContainerRenderer(mxtServletContext, "mapxtreme", req,
    res);

// Create the ImageRequestComposer
ImageRequestComposer imageRC = new ImageRequestComposer(
    mapj, 256, Color.blue, "image/gif");

// Render the image
isRenderer.render(reqEnv);
}
catch(Exception e)
{
// Take appropriate error handling steps
}
```

12 访问远程数据

使用 MapXtreme Java 可从远程资源访问数据以进行地图渲染和分析。本章介绍如何使用命名连接以及连接池这一检索远程数据的有效方式。

本章内容：

命名连接

命名连接是一个资源，它使用别名说明到 JDBC 数据库的连接。这些连接可以预先启动并通过 MapXtremeServlet 汇成连接池，以便客户机应用程序随时使用。

要点：我们强烈建议您实施命名连接和连接池，这样将比为每个客户机请求创建单独连接更加高效。

说明命名连接的信息存储在名为 miconnections.properties 的文件之内。有关信息，请参阅第 217 页的*如何创建命名连接*。

命名连接的安全受益

连接池除了高效率之外，还提供了重要的安全受益。对于三层部署，存储 JDBC 连接信息的 miconnections.properties 文件仍然保存在服务器之上，并且仅限由 MapXtremeServlet 使用。客户机通过别名访问 JDBC 连接，说明连接的用户名和口令等敏感信息不会经由网络传送。

命名连接池

建立到远程数据库的 JDBC 连接可能会消耗大量时间和资源。将这些开支最小化的标准方式是使用命名连接池。在连接池配置方案中，将会创建一组命名数据库连接，然后在众多用户之间复用和共享。

MapXtreme 中的连接池工作机制

通常，连接池将用于 MapXtremeServlet 的服务器端。此外， MapJ 也可配置使用连接池。无论使用哪一方式，其行为均相同。

当需要使用 JDBC 连接从远程数据库访问数据时，将尝试从连接池检索连接。如果连接池中有可用连接，该连接将提供用于相应的应用程序。否则，将会创建新的连接。在使用该连接的任务完成之后，必须将该连接返回到连接池。

MapXtremeServlet 的连接池

MapXtremeServlet 将在其 classpath 中发现 **miconnections.properties** 文件时使用连接池 （在其父级 servlet 容器中的上下文的 classpath）。此文件列出了可用的命名连接。miconnections.properties 文件中的任意连接均可配置为在 MapXtremeServlet 的初始化方法中预先启动。这一点确保了相应连接准备就绪，随时可用于第一个访问 MapXtremeServlet 的客户机。

当连接池由 MapXtremeServlet 使用时，所有打开的连接都将在 MapXtremeServlet 为 servlet 容器所破坏时自动关闭。打开的连接在连接达到预定义的失活时间之后也会关闭，该时间在 miconnections.properties 文件中配置为超时设置。

MapJ 的连接池

如果 miconnections.properties 文件位于使用 MapJ 的应用程序的 classpath 之中，则连接池可以自动在客户机层创建。每个应用程序只能创建一个连接池，该连接池将由在该应用程序之内创建的所有 MapJ 实例所利用。 MapJ 客户机在以下情况下需要访问远程数据源：1) 在进行本地数据访问的图层上执行搜索方法； 2) 在进行本地数据访问的图层获取元数据信息；以及 3) 进行本地渲染。如果连接池准备就绪，则可用于上述各个任务。

如何创建命名连接

命名连接的说明信息包含在名为 **miconnections.properties** 的文件之内。相应信息中包括用于轻松引用数据源、数据源名称、主机、端口、用户名或口令等的别名或昵称。在此文件之中可以设置多个 JDBC 连接。

命名连接可以使用 MapXtreme Java 管理器中的 “连接管理器”面板创建，也可以通过在文本编辑器中手动编辑 miconnections.properties 文件来创建。

创建之后，将 miconnections.properties 文件置于 classpath 之中，以便 MapXtremeServlet 可以在调用启动方法时预先启动相应连接。

miconnections.properties 文件示例

以下各行展示了可见于 miconnections.properties 文件的示例条目。

```
Connection1_name=Pantheon
Connection1_driver=oracle.jdbc.driver.OracleDriver
Connection1_url=jdbc:oracle:thin:@hostmachine:port:sid
Connection1_user=mapxtreme
```

```
Connection1_password=secret
Connection1_is_xy=false
Connection1_prestart=4
Connection1_max=15
Connection1_timeout=300
Connection1_defaultRowPrefetch=75
```

第一行指定了连接的名称。客户机将使用此名称从连接池获取连接。

接下来的 4 行指定了建立 JDBC 连接所需的标准信息：所用 JDBC 驱动程序、数据库的连接 URL、用户名和口令。

再下面一行通知 MapXtreme Java 入池连接所属的数据源是否包含空间对象或空间对象的 X 和 Y 列。

注： MapXtreme Java 不能使用单一命名连接来访问 X,Y 和空间对象数据。

接下来的 3 个设置用于管理要预启动的连接数、连接池可以存储的最大连接数，以及连接关闭并且其资源返回到应用程序之前，在不加使用的情况下可以保持的连接时间。

有关特定数据库的特定附加设置将会出现在列表的末端。例如，以上所示的 Oracle 连接具有 Oracle 的特定键值，名为 defaultRowPrefetch。数据库相关的键值必须是数据库连接可以理解的名称 / 值对。

连接管理器

MapXtreme Java 管理器的“连接管理器”面板提供了管理 JDBC 连接的用户界面（如编辑 miconnections.properties 的内容）。

要从“连接管理器”测试 JDBC 连接，务必确保 JDBC 驱动程序位于 classpath 之中。

“连接管理器”将从 miconnections.properties 文件初始化命名连接的列表。此时既可创建新的连接，也可编辑或删除现有连接。



"连接管理器"的 "编辑"对话框给出了 3 个选项卡用于提供信息。"一般"选项卡收集了连接名、 JDBC 驱动类、用户名和口令。此外，还提供了 "测试连接"按钮，以用于确保连接正常 （如果在 classpath 中没有适当的驱动程序，则测试将会失败）。

在 "自定义"选项卡中，可设置定制属性和值，例如在第 217 页示例中使用的 defaultRowPrefetch。"连接名"选项卡中包含了预启动的连接数和最大许用连接数，以及空闲连接的超时时间。





访问入池连接

MapJ 中的单独图层对象将以特定方式创建，以利用入池的命名连接。所有用于 JDBC 的图层 DataProviderHelpers 均共享公共的构造器类型，取用以下输入参数：

- 字符串 URL
- 属性 connectionProps （用户、口令、预取等）
- 字符串 driverClassName

对于从连接池中检索命名连接的图层，必须使用此形式的 DataProviderHelper 构造器并遵循特殊的命名约定。连接 URL 必须采用以下格式：

```
jdbc:mipool:connection_name
```

在通过别名引用数据源时，其他输入参数 （连接属性和驱动程序名）将被忽略并且应该为空。例如，要连接设置为 Pantheon 命名资源的 Oracle 数据源，需要使用以下信息：

```
OraSoDataProviderHelper oraDpHelper = new
OraSoDataProviderHelper("jdbc:mipool:Pantheon", null, null);
```

有关使用 DataProviderHelpers 的连接池的更多代码示例，请参阅 Javadoc 中的代码示例链接 “连接池 URL 示例”。

管理命名和直接数据库连接

使用 MapXtreme 创建的应用程序可以和 MapXtreme 共享 JDBC 连接。此前，如果应用程序需要直接访问连接，将需要创建和管理其自己的连接集。这些直接的连接应该与由 MapXtreme 维护的任意连接共存。这将导致过量占用资源并且效率低下。

管理连接

现在可通过 **ConnectionPool** 接口创建和管理两种类型的连接。为此可编写此接口的实施，并在系统属性 com.mapinfo.connpool 中确定类名。Mapxtreme 将使用此类来获取其数据库连接 （两种类型：命名连接和直接连接）。

当 ConnectionPool 接收到请求之后，命名连接将从连接池检索 （假定 miconnections.properties 文件位于 classpath 之中）。如果请求的是直接连接，则该连接将从头开始创建。

有关 ConnectionPool 接口的详细信息，请参阅位于 com.mapinfo.dp.conn 文件包中的 MapXtreme Java 文档。

命名连接和直接连接之间的区别

命名连接将在用户通过并为下一请求准备就绪之后，返回到连接池中。直接连接将在不再需要时废弃。

命名连接使用以下格式的别名请求：jdbc:mipool:resource_name。直接连接需要每个请求提供完整的连接信息 （URL、连接属性和驱动程序名称）。

13

图元和搜索

本章介绍 Feature 对象和可以对图元执行的搜索等操作。

本章内容：

Feature 对象

地图图元是地图上的一个地图对象，如点、线或区域等。例如，世界地图可以包含作为国家的区域、作为高速公路的线条以及作为城市的点。在 MapXtreme，地图图元表示为 Feature 对象。例如，英国是区域类型的 Feature 对象， A10 高速公路是直线类型的 Feature 对象，伦敦是点类型的 Feature 对象。

使用数据库的用户对于记录都不会感到陌生。记录是一组相关的信息列。例如，客户数据库中对于每个客户均有一条记录，其中包含了姓名、地址、爱好等列。图元只是一条包含制表数据和几何信息的记录。例如， MapXtreme 示例数据中的 World.tab 文件就是一个 MapInfo 格式的数据库。对于每个国家，均有一条记录与之相对应。每个记录包括若干制表数据列，以及一个指向描述每个国家形状和位置的地理信息的引用，这样一条记录即可显示在地图之上。制表数据也称为属性数据，而几何数据也称为几何对象。这两种类型的数据构成图元。下图进一步阐述了图元的概念：

国家	首都	**Pop_1994**	**Gr_Rt**	**Pop_Male**	几何对象
中国	北京	1,136,429,638	2.2	584,836,207	
墨西哥	墨西哥城	81,249,645	2.2	39,893,969	
美国	华盛顿特区	257,907,937	0.8	125,897,610	

Feature 对象的方法

Feature 对象的方法包含有关制表和几何数据的信息。下表对列出了这些方法：

方法	说明
getAttribute	获取赋予列索引的指定属性。
getAttributeCount	获取与此图元相关的属性数量。
getGeometry	获取相关的几何对象，如果图元没有几何对象则为空。

方法	说明
getLabelRendition	获取为此图元的标注指定的样式。如果没有用于该图元的样式，则返回为空。
getPrimaryKey	获取用于此图元的 PrimaryKey 对象 （唯一 ID）。如果该图元没有 PrimaryKey，则将会返回空值。
getRaster	如果存在，则返回与该图元关联的栅格对象，如果图元没有栅格则返回为空。
getRendition	返回此图元的样式。如果没有用于该图元的样式，则返回为空。

Attributes

每个图元均有一个或多个 Attribute 对象。Attribute 对象表示用于该图元的制表数据列。此对象包含类型和值信息。例如，某一属性为双精度型，其值为表示增长率的 2.2。

Geometries

每个图元具有一个 Geometry 对象。Geometry 对象可用于访问该图元的的所有几何信息。相应的几何信息可为 VectorGeometry 或 PointGeometry。VectorGeometry 用于折线或区域图元。PointGeometry 用于点。

缓冲

MapXtreme Java 提供了创建缓冲区的新方法。从任意输入几何对象 （点、线或区域）均可创建缓冲区。该方法将取用要缓冲几何对象、提供缓冲距离的距离对象以及指定分辨率的整数作为输入参数。返回的输出始终为 VectorGeometry。相应语法如下所示：

```
public static VectorGeometry buffer(Geometry geom, Distance dist,
    int resolution) throws Exception
```

可为此操作提供缓冲距离 （标量值和线性单位）。任意坐标系中的几何对象 （球面坐标系和笛卡尔坐标系）均可缓冲。此外还可以指定相应操作的分辨率，以及在初始几何对象中用于拟合每个点周围的弧的点数。有关详细信息，请参阅联机 Javadocs 中的 GeometryUtils。

Feature 对象和标注样式

每个 Feature 对象可有一个 Rendition 对象。Rendition 对象描述图元的显示特征。Feature 对象可以只取用现有样式信息。相应信息描述 Feature 对象的显示方式。要更改现有 Feature 对象的样式信息，应该使用专题，例如 OverrideTheme 对象。与此类似， Feature 对象可以具有标注样式。

Raster 对象

每个 Feature 对象可有一个 MIRaster 对象。当栅格图像与 Feature 对象相关时，即可检索描述图像的二进制信息。当对象具有栅格时，该对象还将具有一个几何对象，此时由该几何对象描述栅格图像的边界。

代码示例： 从图元获取信息

以下实例代码展示如何使用某些方法来获取有关图元的信息。另请参阅 Javadocs 中的 Feature 主题。

```
List columns = new ArrayList();
Feature ftr;
Geometry geom;
DoubleRect rect;
DoublePoint dblPnt;
PointGeometry pntGeometry;
VectorGeometry vectorGeometry;
PointList pntList;
Attribute attrib;
int attribCount;

// Get the Table information from the FeatureLayer
TableInfo tabInfo = m_Layer.getTableInfo();

// fill vector with Column names
for (int i=0;i<tabInfo.getColumnCount();i++)
{
    columns.add(tabInfo.getColumnName(i));
}

// Perform a search to get the Features(records)from the layer
RewindableFeatureSet rFtrSet = new
    RewindableFeatureSet(m_Layer.searchAll(columns, null));

// get the first attribute
ftr=rFtrSet.getNextFeature();

// then loop through all features in the layer
while (ftr!=null)
{
    /* get the first attribute (columnData) from the feature Note: If
    you want to re-use the Attribute object later on (after the
    getNextFeature loop), you would need to make a copy of the
    Attribute object, using the copy constructor.
    */
    attrib = ftr.getAttribute(0);
```

```
// get a count of all attributes in the layer
attribCount = ftr.getAttributeCount();

// get the reference to the geographic information from
// the feature
geom = ftr.getGeometry();

// check to see if the geographic object is a Point
if (geom.getType()==Geometry.TYPE_POINT)
{

    // Cast the general geometry to a point geometry
    pntGeometry = (PointGeometry)geom;

    // get the minimum bounding rectangle for the feature
    rect = pntGeometry.getBounds();

    // get the x,y location where the feature 調 label will be
    // anchored
    dblPnt = pntGeometry.getLabelPoint(null);
}
else
{

    // Cast the general geometry to a Vector geometry
vvectorGeometry = (VectorGeometry)geom;

    // get the minimum bounding rectangle for the feature
    rect = vectorGeometry.getBounds();

    // get the x,y location where the feature 調 label will be
    // anchored
    dblPnt = vectorGeometry.getLabelPoint(null);
    double[] pnts;
    int offset=0;
    int numPts;

    // Loop through all the point groups and then put the
    // points into an array
    for (int i=0;i<vectorGeometry.getPointListCount();++i)
    {

        // Get the next Point List
        pntList = vectorGeometry.getNextPointList();

        // determine the number of Points in the point group
        numPts = pntList.getPointCount();

        // Create the point array large enough to hold all the
```

```
                // points

                pnts = new double[numPts];

                // Call getNextPoints which will put the points into the
                // array
                pntList.getNextPoints(pnts, offset, numPts/2);
            }
        }

        // Get the next feature
        ftr=rFtrSet.getNextFeature();
        }

    // Rewind the FeatureSet to prepare for future use
    rFtrSet.rewind();
```

使用 FeatureFactory 创建图元

MapXtreme 可用于创建、修改或删除图元 （点、线、折线、区域），也可用于将这些对象添加到注释图层、 TAB 或 JDBC 图层。新建地图图元有两种途径。既可以使用 FeatureFactory 对象来创建图元，也可以通过使用 Layer 类的搜索方法检索现有图元来创建新图元。

FeatureFactory 方法

FeatureFactory 对象的方法可用于新建表示点、线、折线和区域的地图图元。这些方法如下所示：

- createPoint
- createPolyline
- createRegion
- createCircularRegion
- createEllipticalRegion

上述方法均返回独立的 Feature 对象。要通过 FeatureFactory 创建任意对象，需要指定 Rendition、 Label Rendition、与该 Feature 相关的 Attributes 数组、用于该 Feature 的几何对象和 PrimaryKey。

注：　对于 TAB 文件而言，将在图元添加到图层时指定 PrimaryKey。此外，主键不能更改。

创建了要添加到 JDBC 表图层的图元之后，该图元将最终将存储为数据库表中的一行。提供给 FeatureFactory 的 Attributes 数组是用于这一新行的列值。这些值必须完全符合从图层的 TableInfo 检索的列名的顺序，即 TableInfo 列名数组中的每个名称均必须在图元的 Attributes 数组中具有相应的值。如果 JDBC 图元 Attributes 数组包含空引用，则相应的列将作为 NULL 值插入到数据库中。对于诸如新 Attribute((Double)null) 之类表示 NULL 值的 Attribute 对象，也可用于上述情况。

图元还需要指定非空的 PrimaryKey 值。但是对于 TAB 文件则无需指定。确保该 PrimaryKey 的值是指定用于相应图元的 Attributes 数组的一部分。

用于 JDBC 图元的几何对象必须位于数据库表的坐标系之中，否则将报错。

要创建点，可指定中点，以及其符号大小、字体和颜色等样式和 Attributes 数组。

为了说明圆的几何结构，除了指定样式和属性之外，还必须使用其他参数，包括分辨率，以及是否使用显示或数字坐标对此圆进行说明。分辨率定义绘制近似多边形时使用的节点数和第一个参数，该参数控制图元为显示或数字圆。可以用屏幕 （显示）的坐标系或地图（数字）的坐标系中绘制圆。在屏幕上，显示版本的外观通常比较精确 （例如，更像一个圆），然而数字版本可能显示为椭圆形，因为它是按照地球的曲线绘制的。

除了用于圆的参数之外，椭圆区域还采用双精度型来说明 x 半径和 y 半径。

创建区域或多边形涉及创建 double point 数组来说明几何对象的操作。

代码示例

此示例说明如何在 FeatureFactory 中创建各个图元类型：点、圆、线、椭圆形和多边形。要创建一个区域，请按照多边形示例来创建传递到 createRegion() 的点数组。

还可以在 FeatureFactory 类的 Javadocs 中找到此代码示例。

```
// Get Feature Factory reference
//map is a MapJ object
FeatureFactory ff = map.getFeatureFactory();

// Set up Attribute object
Attribute att[] = new Attribute[1];
att[0] = new Attribute("Feature1");

// Set up rendition object
/* This will not work for tab files. For tab files you must use the
com.mapinfo.tab.TABStyleFactory to create an appropriate Rendition
object.
*/
Rendition rend = RenditionImpl.getDefaultRendition();
```

```
// For circles, specify the edge and fill color.
rend.setValue(Rendition.STROKE, Color.cyan);
rend.setValue(Rendition.FILL, Color.green);

// For points, specify the symbol size, font, and color
rend.setValue(Rendition.SYMBOL_STRING, "@");
rend.setValue(Rendition.FONT_SIZE, 16);
rend.setValue(Rendition.FONT_FAMILY, "MapInfo Shields");
rend.setValue(Rendition.SYMBOL_FOREGROUND, Color.blue);

// For lines, specify the line color and width
rend.setValue(Rendition.STROKE, Color.green);
rend.setValue(Rendition.STROKE_WIDTH, 4);

// For ellipses, specify the fill color and opacity
rend.setValue(Rendition.FILL, Color.blue);
rend.setValue(Rendition.FILL_OPACITY, new Float(0.40));

// Set the center point features for the circle
DoublePoint dp = new DoublePoint(-104, 45);

// Create Circular region
int circType=1;
int circRadius=25;
int circResolution=25;

// For Elliptical region
// Create an integer variable to set the coordinate system of the
// region. 0 sets for map coords, 1 sets screen coords
int type = 1;

// Create a DoublePoint variable to set the center of the ellipse
DoublePoint centerPt = new DoublePoint(-73.702000, 42.682599);

// Create a double variable to set the rotation angle of the ellipse in
// radians
double xRadius = 1.0;
double yRadius = 0.25;

// Create an integer variable to specify the number of points that will
// describe the ellipse. If 0, the default of 12 will be used.
int resolution = 0;
Feature retFeature;
try
{
    // Create an Annotation layer for storing the features we're
    // creating
    AnnotationTableDescHelper annTableDesc = new
    AnnotationTableDescHelper("Annotations");
```

```
        AnnotationDataProviderHelper dpHelper = new
        AnnotationDataProviderHelper();
        LocalDataProviderRef dpRef = new LocalDataProviderRef(dpHelper);
        Layer annotLayer = map.getLayers().insertLayer(
            dpRef, annTableDesc, 0, "Annotations");

        //PrimaryKey is taken as an argument by all the create methods and
        //cannot be null
        PrimaryKey pkey = new PrimaryKey(att[0]);

        // Create a Circular Region
        retFeature = ff.createCircularRegion(circType, dp, circRadius,
        LinearUnit.mile,circResolution, rend,null, att,pkey );

        // Add the new feature to the annotation layer
        PrimaryKey pk = annotLayer.addFeature(retFeature);

        // Create Point
        retFeature = ff.createPoint(dp, rend,null, att, pkey);
        pk = annotLayer.addFeature(retFeature);

        // Create PolyLine using a 2 x 6 double matrix
        // Row 1 contains 3 points(x,y,x,y,x,y) and 2 line segments.
        // Row 2 contains 3 points(x,y,x,y,x,y) and 2 line segments.
        double pts[][] = { {-104, 45, -102, 46, -100, 45},
            {-100, 45, -98, 44, -96, 46} }
        retFeature = ff.createPolyLine(pts, rend, null, att, pkey);

        //Create Elliptical Region
        retFeature = ff.createEllipticalRegion(type,centerPt,angle,
        xRadius,yRadius,LinearUnit.mile,resolution,rend,null,null,pkey);
        pk = annotLayer.addFeature(retFeature);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
```

FeatureSet 集合

FeatureSet 是 Feature 对象的集合。在 MapXtreme 中，构成地图的各个图层通常在每个图层中都拥有相同的图元类型。例如，“世界”图层拥有代表每个国家的区域图元，“高速公路”图层拥有代表美国主要高速公路的线图元，而 “世界首都”图层则拥有代表每个国家首都的点图元。 Layer 对象的搜索方法从图层中返回 FeatureSet 集合。

注：　　集合中的图元不按特定的顺序显示。

使用以下方法可控制 FeatureSet 对象：

方法	说明
dispose	处理由 FeatureSet 使用的资源。使用 FeatureSet 完成操作后，将调用此方法。
getNextFeature	获得集中的下一个图元。
getRendition	获取此 FeatureSet 中所有图元的基础样式。
getTableInfo	获取说明此 FeatureSet 的 TableInfo （元数据）。
isRewindable	决定此对象的可重绕状态。
rewind	在第一个图元之前重绕 FeatureSet。

为了将内存分配降至最低，在从 getNextFeature 方法返回图元时， MapXtreme 可能复用同一内部内存。如果需要在下一次调用 getNextFeature 之后保留所有或部分图元，则制作要持续使用的对象副本。这就意味着 FeatureSets 只能正向移动，只要传递了一个功能就无法返回到它。

一些 FeatureSets 可能是可重绕的，这就意味着 FeatureSet 可以重置为它的第一个图元。这是每个数据提供方的实施细节。如果 FeatureSet 不可重绕，那么可从不可重绕的 FeatureSet 创建可重绕的 FeatureSet，然后再重新移动一次 FeatureSet。

以下是重绕 FeatureSet 的示例：

```
if(!fs.isRewindable() )
{
    fs = new RewindableFeatureSet(fs);
}
```

使用 FeatureSets 完成操作后，将始终调用 dispose 方法。

搜索

MapXtreme 最强大的功能之一是搜索功能。使用搜索功能可以按照地理信息检索特定的数据。例如，如果要查找 25 米半径范围内的所有基站，那么将执行搜索操作。

搜索是 Layer 对象的方法。它们返回 FeatureSet 对象。MapXtreme 的基本功能是选择地图上的图元，以便可针其执行其他任务。用户可以单击地图来选择一个或多个图元 （点、线、区域等）。搜索结果通常解释为选择内容。

以下 Layer 对象的方法提供了各种搜索图层并返回 FeatureSet 集合的方式。

- searchAll
- searchWithinRadius
- searchWithinRegion
- searchWithinRectangle
- searchAtPoint
- searchByAttribute
- searchByAttributes

搜索搜索操作都将传递要返回的列的名称和查询参数。从任何搜索操作返回的列的名称应该放在 List 对象中。

限制搜索操作返回的信息

从搜索操作返回的图层上图元的特征取决于几种可选参数。默认情况下，将使用查询返回与图元相关的几何对象、样式、标注样式、首选标注位置和栅格数据。如果您想限制某个图元返回的信息，可以使用 QueryParams 类。这样会改善查询性能。

QueryParams 类还指定查询的 SearchType。从查询返回的图元取决于指定作为查询一部分内容的搜索类型。使用 *mbr* 搜索类型的查询返回其最小边界矩形和搜索区域交叉的图元。此搜索类型的限制条件最少，却可以返回最多的图元。

使用 *partial* 搜索类型的查询返回与搜索区域交叉的图元。

使用 *entire* 搜索类型的查询返回完全包含在搜索区域中的图元。这是限制条件最多的搜索类型。如果您未使用 QueryParams，则 SearchType 默认是 *mbr*。

部分和全部都是绝对的，有效的搜索比较图元的真实几何范围和搜索区域。在需要真实和精确的结果时，必须使用这些搜索类型。 mbr 搜索是一个快捷的近似方法，简化了几何对象，允许更加快速的比较。在例如 Oracle Spatial 的特定数据源中， mbr 搜索 (SDO_FILTER) 实际上是通过将图元的最小边界矩形和区域的最小边界矩形交叉来实施的，这样可以产生的 “命中数”比使用实际搜索区域更高。

此示例显示 QueryParams 对象是如何限制搜索操作所返回的信息：

```
// find all Features entirely within a given search region, return a
// single Attribute column and no Rendition information.
List cols = new ArrayList();
cols.addElement(褻ounty”);
```

```
Feature searchFeature =
    mapj.getFeatureFactory().createRegion(points, rend, attribs,null);
QueryParams queryParams = new QueryParams(SearchType.entire, true,
false, true, true, false);
FeatureSet fs =
    layer.searchWithinRegion(
        cols, searchFeature.getGeometry(),queryParams);
```

注： 由于在实施 SQLServer Data Provider 过程中存在某些限制，因此如果 SearchType 不是 mbr，则 searchInRegion() 请求可能会降低性能。当要搜索的搜索区域和几何对象是较大的复杂区域时，这一点尤为明显。

GML 格式的向量响应

MapXtreme Java 提供了以 GML 格式从搜索操作获取向量响应的能力，这种格式是说明地图对象的几何范围的 XML 编码。

可以由 MapJ 客户端应用程序和非客户端使用 MapVectorRequests。对于从 MapJ 类创建的客户端，将利用新的 **VectorRequestComposer** 类，通过编程方式构建请求。此类包含在 MapJ 图层对象和命名图层上所有支持的搜索方法的工厂方法。

对于非 MapJ 客户端应用程序，必须通过开发人员自己的途径，按照控制 XML 文档内容的 DTD 语法和其他策略来创建 MapVectorRequest 文档。

除非传递到 VectorRequestComposer 的 create*Feature*() 方法的 QueryParams 对象配置为不返回几何对象，否则 MapVectorRequest 将返回几何对象，其中 *Feature* 代表点、多边形或区域。

代码示例

有关在搜索过程中如何使用 VectorRequestComposer 的代码示例，请参阅 VectorRequestComposer 类的 API 文档中引用的联机示例。

搜索方法

本节定义了可用的各种搜索方法和展示其用法的代码。

searchAll

返回图层中所有图元的 FeatureSet 集合。如果应用程序需要循环整个图层，可使用此搜索方法。

```
//Assume fs is a FeatureSet object.
//Assume columnNames is a List of the columns to be
//returned.
//Assume qp is the QueryParams object.
//Assume myLayer is a FeatureLayer object.
try {
        fs = myLayer.searchAll(columnNames,qp);
}
catch(exception e) {
        e.printStackTrace();
}
```

searchWithinRadius

返回在点对象的指定距离之内构成图元的 FeatureSet 集合。使用此搜索可查找离指定位置最近的经销商，或者返回一家商店某一半径范围内的客户数据。

```
//Assume fs is a FeatureSet object.
//Assume columnNames is a List of the desired columns
// to be returned.
//Assume qp is the QueryParams object.
//Assume myLayer is a FeatureLayer object.
DoublePoint dblPt = new DoublePoint(-73.889444,42.765555);
double dRadius = 10.03;
try {
fs = myLayer.searchWithinRadius
        (columnNames,dblPt,dRadius,LinearUnit.mile,qp);
}
catch(exception e) {
        e.printStackTrace();
}
```

searchWithinRegion

此搜索方法返回一个图元的几何范围内构成图元的 FeatureSet 集合。使用此方法可返回特定区域内的客户数据（如邮政编码），或返回用 FeatureFactory 创建的区域内包含的图元。

```
private boolean layerSearchWithinRegion()
{
    //Assume fs is a FeatureSet object.
    //Assume columnNames is a List of the columns to be
    // returned.
    //Assume qp is the QueryParams object.
    //Assume myLayer is a FeatureLayer object.
    //Assume vGeom is a VectorGeometry of TYPE_REGION
    try {
        fs = myLayer.searchWithinRegion(columnNames,vGeom,qp);
}
catch(Exception e) {
    e.printStackTrace();
    return false;
}
    return true;
}
```

searchWithinRectangle

此搜索方法返回指定矩形界限内的 FeatureSet 集合。使用此方法可在给定地图窗口内进行搜索或预检缩放级别，以查看它是否包含某些要关注的点。

```
//Assume fs is a FeatureSet object.
//Assume columnNames is a List of the columns to be
// returned.
//Assume qp is the QueryParams object.
//Assume myLayer is a FeatureLayer object.
DoubleRect dRect =
    new DoubleRect(-74.092662,42.765555,-73.668898,42.856420);
try {
    fs = myLayer.searchWithinRectangle(columnNames,dRect,qp);
}
catch(exception e) {
    e.printStackTrace();
}
```

searchAtPoint

此搜索方法返回构成指定点上的图元的 FeatureSet 集合。使用此方法可以测试与某个点交叉的所有对象。使用此方法可以测试给定的位置是否位于某个覆盖范围之内。

```
//Assume fs is a FeatureSet object.
//Assume columnNames is a List of the columns to be
// returned.
//Assume qp is the QueryParams object.
//Assume myLayer is a FeatureLayer object.
DoublePoint dp = new DoublePoint(12.3456,-67.890);
```

```
try {
    fs = myLayer.searchAtPoint(columnNames,dp,qp);
}
catch(exception e) {
    e.printStackTrace();
}
```

searchByAttribute

此方法返回其属性与给定属性相匹配的 FeatureSet 集合。使用此方法可以选择带有公用属性信息的所有功能。例如，如果拥有一张包含家庭收入列的表，即可使用 searchByAttribute 来返回家庭收入超过 100,000 美元的所有记录。

```
// Assume fs is a FeatureSet object.
// Assume columnNames is a list of the columns to be
// returned.
// Assume myLayer is a FeatureLayer object.

try {
    // return Features where "Annual_Income" equals $100,000

    Attribute mySearchAttr = new Attribute(100000);
    String searchCol = "Annual_Income";

    fs = myLayer.searchByAttribute(
        columnNames, searchCol, mySearchAttr, null);
}
catch(Exception e) {
    e.printStackTrace();
}
```

searchByAttributes()

本版本的 MapXtreme Java 为调用 **searchByAttributes()** 的图层提供了新的搜索方法。使用此方法可以比较几列中的值，并返回匹配标准的图元。将使用等号运算符比较当前的值。

此方法将取代 searchByPrimaryKey （现已过时）。定义关键字的列现在包含在 searchByAttributes() 的 attNames 参数中。代表关键字的值将放在 attValues 参数的 AttTuple 对象中。 AttTuple 对象替换主键，作为值的持有者。

采用 searchByAttributes()，当 FeatureSet 中的一个图元满足任何 AttTuple 对象的搜索条件时，会将其返回。如果使用 OR 运算符计算的多个 AttTuple 对象隐含条件，也可返回图元。

searchByAttributes() 的语法如下所示：

```
public FeatureSet searchByAttributes (List columns, List attNames,
List attOperators, List attValues, QueryParams queryparams)
```

```
throws Exception {...}
```

搜索方法的常规形式如下：

"return the requested columns from all features where (colA = v1 and colB = v2) OR (colA = v3 and colB = v4) OR ...(colA = v9 and colB = v10)".

定义参数的方式如下：

```
attNames = {colA, colB}       // string objects
attOperators = {"eq", "eq"}   //AttOperator objects
attValues = {v1, v2), (v3, v4), ....(v9, v10)}
   //AttTuple objects that contain Attribute objects
```

列需要满足以下两个条件的任意列：一是其所在的表中可使用简单运算符，二是其值可以用 Attribute 类表示。

可返回空列表，但不能是空值。

由于 searchByPrimaryKey() 已过时，因此可以使用 QueryParams 接口上的新构造器。无需在 QueryParams 中包含键列，它们现已包含在由 searchByAttributes() 中的 attNames 定义的列列表中。

对于编写自定义查询的用户来说，请注意此前使用 PrimaryKey 的 QueryBuilder 接口中的方法已经过时，不再支持 **queryByAttributes**。

以下代码示例展示了搜索其 Capital 字段中的值等于 Albany 的图层中的所有图元的过程。

```
/* @param lyr the FeatureLayer to search on.
    * @param columns A java.util.List of columns to be returned
    * @return FeatureSet FeatureSet containing the features that match
    * the search criteria. May be empty
    */
    public FeatureSet searchByAttributes(
        FeatureLayer lyr, List columns)
    {
        FeatureSet ftrSet = null;
        String mySearchColumn = "Capital";
        String mySearchAttr = "Albany";
        try {
            // create a search Attribute
            Attribute mySearchAttr = new Attribute(mySearchAttr);
            // create a List of columns to search on
            List namesList = new ArrayList();
            namesList.add(mySearchColumn);
            // create a List of column values to search for
            List valuesList = new ArrayList();
            AttTuple myTuple = new AttTuple(mySearchAttr);
            valuesList.add(myTuple);
            // create a List of search operators (this example uses
```

```
            //'equal')
            List operatorsList = new ArrayList();
            operatorsList.add(AttOperator.eq);
            // do the search
            ftrSet = lyr.searchByAttributes(columns, namesList,
            operatorsList, valuesList, QueryParams.m_defaultQP);
            return ftrSet;
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }
```

搜索由 SQL 查询定义的图层

MapXtreme Java 执行用户定义的 SQL 查询，并无需对此查询进行任何修改。 MapXtrem 将在写入查询时执行它 （该查询称为 “传递” 查询），并检索图层中的所有图元。该查询可以返回很多不显示的图元，例如在缩小图元密度较高的图层的情况下。

传递查询用于更高级的 MapXtreme Java 用户，这些用户需要自定义查询来构造图层数据，并且了解如何包含相应的限制条件。

QueryBuilder 接口

为了帮助技术型用户通过传递查询制定限制条件， MapXtreme Java 提供了一个接口，您可以使用这个接口写入您自己的回调对象，以在由传递查询定义的图层上进行渲染或执行搜索时，创建修改的查询字符串。 QueryBuilder 对象将指定到传递图层，其在需要时调用它的方法。

在地图渲染过程中，如果 MapXtreme Java 遇到由包含 QueryBuilder 的传递查询定义的图层，将调用 QueryBuilder 方法 **queryInRectangle**，以提供传递到渲染器的查询字符串。QueryBuilder 提供了构造新查询字符串所需的全部数据，这个新的查询字符串中包含的限定性几何条件将返回的图元仅限于显示视口中可见的图元。如果此图层没有 QueryBuilder，那么渲染它时返回的图元会远远多于显示的图元，从而严重影响了效率。（对于没有通过使用 INFO 级别或更高级别的 com.mapinfo.util.Logger 类记录输出渲染的返回图元，您可以确定其数量。）

利用调用任何搜索方法搜索传递层的操作需要通过添加 where 子句和 / 或更改 select 子句中的列修改查询。每种搜索方法都在 QueryBuilder 接口上调用它的对应方法，并使用新的查询字符串来执行搜索。没有 QueryBuilder，传递图层搜索将会报错。 QueryBuilder 必须应用于任何由传递查询创建的图层才能执行搜索。

要在 Layer 对象上设置 QueryBuilder，请遵循以下示例：

```
Layer.setQueryBuilder(QueryBuilder myQB);
```

QueryBuilder 的考虑因素

- QueryBuilder 接口是技术型用户使用的功能，只能在图层的表定义不足的情况下使用。
- 使用 QueryBuilderXMLHandler 接口，可以从地图定义保存 / 存储新的 QueryBuilder 引用。
- QueryBuilder 接口只能在客户端应用程序中使用，也就是说，在哪里进行数据访问就在哪里驻留 MapJ 实例。不会将 QueryBuilder 对象发送到服务器。
- 可以将同一 QueryBuilder 引用用于多个图层。
- 从 QueryBuilder 返回的查询将采用与所有传递查询完全相同的方式执行。
- 使用 QueryBuilder 并不能更改定义图层的任何数据。返回的查询执行一次后就会废弃；它不会替换用于构造 Layer 对象的 TableDescHelper 中的原始查询。
- 通过 QueryBuilder 查询返回的数据拥有的主键定义、维数、坐标系和空间列（如果有的话）必须与在 TableDescHelper 中最初确定的值相同。（可能会在以后的版本中解除 QueryBuilder 的这种限制。）

示例代码

在 MapXtreme Java 的 /examples/client/QueryBuilders 目录中提供的是 OracleQueryBuilder 的实施示例， Oracle 用户可以将此作为入门学习范本。

此外还可以找到 IdentityQueryBuilder 的示例代码，其返回未更改的原始输入查询。这对使用基础类开发新 QueryBuilder 尤为实用。

图元编辑

MapXtreme 可用于添加、修改或删除构成注释图层的图元（点、线、区域等）， Annotation 图层是通过 JDBC 数据源中的表或 TAB 图层置入图层的。使用以下 FeatureLayer 类方法可以完成这些操作：

- addFeature

- addFeatureSet
- replaceFeature
- removeFeature

添加到图层中的图元可以在 FeatureFactory 中创建或成为执行搜索的结果 （searchWithin、searchBy 等），有关 FeatureFactory 的信息请参阅第 228 页的*使用 FeatureFactory 创建图元*，有关搜索图层的讨论，请参阅第 232 页的*搜索* 。

只能在客户端应用程序中执行图层编辑。请务必使用 LocalDataProviderRef 创建计划或可能需要编辑的所有图层。

编辑注释图层

注释图层包含在特定地图区域上标记或放置重点的图元。注释图层不与任何持久的数据源相关联，因此对注释图层所做的更改只会影响当前 MapJ。编辑注释图层只更改为该图层渲染的图像。

注释图层图元的主键

添加到注释图层的图元必须拥有非空的主键值，其由一个或多个还指定为图元 （非空） Attributes 数组的一部分的属性值构成。

继第一个图元之后陆续添加到注释图层的图元必须与第一个图元的主键结构相匹配 （定义主键值的属性值的数据和类型相同）。

注释图层图元的坐标系

对于注释图层，图元应该位于 MapJ 对象的数字坐标系中。当使用 FeatureFactory 创建图元时，有必要指定 MapJ 数字坐标系中的输入坐标数组。当采用通过搜索方法返回的 FeatureSet 中的图元时，图元已位于 MapJ 数字坐标系中。

编辑 JDBC 表图层

可以由数据库表名或数据库传递查询定义 JDBC 图层。然而，只能编辑由数据库表名定义的图层（添加、替换或删除图元），因为对图层所做的更改实际是对源数据库表进行的更改。

对该表所做的更改必须符合在表的方案定义中定义的任何约束条件。例如，某些列可能需要是非空、唯一形式；拥有某一范围之内的数字值；拥有大于零的数字值或拥有某一长度范围内的字符串值。违反这些约束条件将造成数据库出错。

此外，还必须具备更改数据库表的权限。

对 JDBC 表图层进行更改的持久性

对 JDBC 表图层成功进行更改会引起源数据库表中发生更改。然后，当下一次从数据库刷新图层数据时，将在 MapJ 中看到所做的更改。 MapXtreme 将每个图元编辑请求视为单独的事务处理，并且在成功完成每个请求之后立即提交对数据库所做的更改（或如果更改失败，则将立即回退）。当请求是 addFeatureSet 时，集合中的每个单独的图元将发生提交（或回退）。

编辑 JDBC 图层中的图元的规则

向 JDBC 图层添加图元时，必须遵循以下规则。在添加 FeatureSet 和替换图元时，这些规则同样适用。有关详细信息，请参阅 Javadocs 中的 FeatureLayer.addFeature()。

图元定义

必须按照通过调用 FeatureLayer.getTableInfo() 返回的 TableInfo 准确定义所有要添加的 JDBC 图元。特别是，图元的 Attributes 数组必须与 TableInfo 的列名数组完全对应，也就是说， TableInfo 列名数组中的每个名称必须在图元 Attributes 数组中拥有相对应的值。如果 JDBC 图元 Attributes 数组包含空引用，则相应的列将作为 NULL 值插入到数据库。对于诸如新 Attribute((Double)null) 之类表示 NULL 值的 Attribute 对象，也可用于上述情况。

图元几何对象

JDBC 图元的几何对象必须位于数据库表的坐标系之中，否则将出错。

图元样式

如果表列的样式类型是 RenditionType.MAPXTREME，则只插入样式；否则将忽略样式。

自动递增列

如果数据库中的任何列是 “只读” 或 “自动递增” 列，即只能通过数据库设置它的值，那么 Mapxtreme 将忽略这些列。这些列类型在一些数据库中用于主键值。所有列 （包括主键列）都必须以 JDBC 图元 Attributes 数组表示。 MapXtreme 不提供缺失的值。如果整数键列需要唯一值，那么可能使用 getColumnStatistics() 来查找列中的最大值，并添加一个值来获取唯一的值。

编辑 Tab 图层

MapXtreme Java 提供了编辑本地 MapInfo TAB 文件的功能。只能更新 Java 有权编辑的 TAB 文件。也就是说，不可编辑标记为只读或由其他用户或组拥有的文件。除非在 Java 安全管理器中授予了特殊的权限，否则也不允许从 applet 修改文件。

添加的图元应该与底层 TAB 文件拥有相同的表结构。

列值必须符合在表中定义的规则。例如，具有字符列或小数精确度列的列。

编辑 Tab 图层中的图元的规则

向 TAB 图层添加图元时，必须遵循以下规则。

图元样式

必须使用 com.mapinfo.tab.TABStyleFactory 类创建图元样式。从而可以使样式与在 MapInfo Professional 中定义的样式保持一致。

图元几何对象

假定所有添加的几何对象与 TAB 文件位于同一坐标系中。

Tab 图层图元的 PrimaryKey

当向 TAB 图层添加新功能时，将自动分配 PrimaryKey。可以通过两种方式返回 TAB 文件的 PrimaryKey。第一种方式是通过使用过时的 QueryParams。第二种方式是通过请求伪列名 com.mapinfo.dp.tab.TABTableDescHelper.KEY_COLUMN_NAME。此列中的属性值可以放在新的 PrimaryKey 实例中。

14

标注和样式

本章介绍如何使用 API 设置各种标注属性和样式。

本章内容：

标注概览

标注元素可通过 MapXtreme Java API 控制。此外，还可以通过 MapXtreme Java 管理器中的“图层控制”对话框上的 “标注”按钮或 Layer Control Bean 来控制这些属性。本章侧重于介绍 API。有关通过图层控制对话框进行标注的详细信息，请参阅*第 5 章：管理 MapXtreme Java* 。

专题标注

标注是基于通过 LabelProperties 对象为其设置的属性来绘制的。在此前的版本中，一个图层只能设置一个 LabelProperties 对象。因此该图层的所有标注的外观均比较相似。

现在通过引入标注专题，每个图层可以具有多个 LabelProperties 对象。和图元专题的概念类似，标注专题在 LabelProperties 对象上工作。但是标注专题的功能比图元专题更加强大，其不仅可以修改标注的 Rendition 对象，还可以影响到标注的位置、优先级、文本和缩放性能。

正如图元专题一样，有多种类型的标注专题可供使用：RangedLabel 专题、IndividualValueLabel 专题、 OverrideLabel 和 SelectionLabel 专题。这些专题的用法和图元专题的用法类似：尤其是在使用不同类型的样式、布放方式、标注文本对标注集合进行操作的时候。

例如，要标注一个图层，该图层的每个标注的外观均由与相应图元关联的值控制。此时可创建 RangedLabel 专题来显示相应图元的标注，对于范围之内高端的标注，可采用与低端范围内的标注不同的颜色或字体大小来表示。

要更改搜索返回的图元的 LabelProperties，可使用 SelectionLabelTheme。这一操作可通过先将 FeatureSet 添加到选择，然后将该选择与 SelectionLabelTheme 相关联来完成。

每个图层可以应用多个标注专题。专题列表顶部的专题可以覆盖排序较为靠后的专题的设置。在 LabelProperties 对象中提供的属性定义了各种标注专题如何一起显示最终的标注特征。

代码示例 : LabelProperties

本例使用默认的位置设置、多行文本，并使用源自表的多列和符合图元路径的标注来进行标注。

```
FeatureLayer thisLayer =
    (FeatureLayer)mapj.getLayers().getLayer("States");
```

```
LabelProperties labelProps = new LabelProperties();

// Set the Label offset and alignments to their default positions.
labelProps.setOffset(LabelProperties.DEFAULT_OFFSET);
labelProps.setHorizontalAlignment(
    LabelProperties.HORIZ_ALIGN_DEFAULT);
labelProps.setVerticalAlignment(LabelProperties.VERT_ALIGN_DEFAULT);

// Turn the multiline option on and set the columns to be used for
// labeling.
labelProps.setMultiLineTextMode(LabelProperties.MULTILINE_TEXT_ON);
labelProps.setLabelExpression(
    "\"NAME:[\"+state_name+\"]\\nABBREV:[\"+state+\"]\"");

// Set LabelFollowingPath to true to enable label splining
boolean bFollowPath = true;
labelProps.setLabelFollowingPath(bFollowPath);

// Set a Rendition for the Labels and create an OverrideLabelTheme in
// order to display the options set.
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.FONT_FAMILY, "Arial");
rend.setValue(Rendition.FONT_WEIGHT, 2);
labelProps.setRendition(rend);
OverrideLabelTheme orLabelTheme = new OverrideLabelTheme(
    labelProps, "Theme Name");
thisLayer.getLabelThemeList().add(orLabelTheme);

// Get the States Layer, set its LabelProperties object, and turn
// labeling on.
BaseLabelProperties base_label = new BaseLabelProperties(labelProps);
thisLayer.setLabelProperties(base_label);
thisLayer.setAutoLabel(true);
```

标注的缩放设置

此前，每个图层只支持一个 LabelProperties 对象，该对象指定了缩放属性。现在，标注专题指定了其自己的缩放设置，图层的每个专题均可有其自己的缩放设置。在 LabelProperties 上的缩放属性已经弃用。此外，进一步增强的图元专题还包括了缩放和可见性设置。

每图元标注样式

地图图元部分上是通过使用诸如颜色、线条宽度、填充图案和符号样式等信息渲染的。这些样式通过调用 Feature 接口的 **getRendition** 方法返回。在 MapXtreme Java 3.x 中引进之后，每个图元即可具有其自己的样式。

每图元标注可用于提供预定义的 Rendition 对象来描述每个图元的标注。在为图层创建适当的 TableDescHelper，提供每图元标注样式列和说明如何解释列数据的信息（如 MapBasic 子句和 XML 格式等）的过程中，将会访问这些预定义的对象。**getLabelRendition** 方法用于返回图元标注的样式属性。有关样式的说明，可参阅自第 259 页页开始的内容。

LabelProperties 类

LabelProperties 类包含控制如何绘制用于每个图层的标注的方法。借助于此类中的方法，可以控制标注的内容、可见性、外观和相对重要性。

标注列

标注文本来自于和地图图元关联的属性。这两个元素是动态链接的。如果底层属性更改，则标注文本也会发生相应的更改。

要控制将哪一属性列用于图层标注，可使用 LabelProperties 类中的 **setLabelColumn** 方法，并按照名称而非索引来指定列。

注： 取用表示列索引的整数的 setLabelColumn 方法现在已经弃用。

例如，为了让地图更加便于用户理解，可以使用入学年龄段的人口取代学校的名称，来标注学校所在街区。

标注表达式

MapXtreme Java 可使用信息列、静态文本或两者的组合来对图元进行标注。要组合列信息和静态文本，需要为标注创建一个表达式。例如，需要为名为 POP_2000 列的实际人口值创建一个采用静态文本 "Pop:" 的标注。可使用以下方法：

```
labelprops.setLabelExpression("Pop:" ??+ POP_2000}
```

使用 setLabelExpression 是最好采用 setLabelColumn，该方法可以对单标注列执行。

标注样式

标注样式涵盖了多种字体外观元素，例如用于标注文本、大小、前景和背景色以及特效的字体。

诸如 Type 1 或 TrueType 之类 Java 2D 平台支持的任意字体均可用于标注。与图层的 LabelProperties 对象关联的样式 （检索自其 getRendition 方法）用于控制字体、字体颜色和大小，以及标注的效果，其中包括下划线、斜体和用于框格的背景色、光晕或轮廓。

例如，作为惯例，首都的标注要比其他城市的标注较大。要在绘制地图时突出这些首都，可能需要使用带有光晕效果的标注，令其有别于周围的其他城市。

本例表示标注文本将更改为红色的粗体斜体文本：

```
//Change the Rendition
LabelProperties labelProp = layer.getLabelProperties();
LabelProperties labelProp = new LabelProperties;
Rendition labelRend = labelProp.getRendition();
labelRend.setValue(Rendition.FONT_WEIGHT, 2f);
labelRend.setValue(Rendition.SYMBOL_FOREGROUND, Color.red);
labelRend.setValue(Rendition.FONT_STYLE, Rendition.FontStyle.ITALIC);
labelProp.setRendition(labelRend);
```

创建可缩放标注

Rendition.FONT_SIZE 取用的距离单位既可以是以纸介单位描述的大小，如毫米；也可以是诸如英里或公里的地理单位。通过使用地理单位，可以创建能够缩放的标注，在地图缩放时适当自行重调大小。在放大地图时，标注也将显示为较大的尺寸。这一特性同样适用于符号大小。

上述代码示例使用地理单位。对于其他示例，请参阅本章位于第 266 页的样式一节。

以下代码示例展示如何设置字体大小：

```
// This example sets a font size of 18 for the layer's labels
Distance distance = new Distance(18, LinearUnit.mile);
labelRend.setValue(Rendition.FONT_SIZE, distance);
labelProp.setRendition(labelRend);
```

多行文本

目前，增强的文本格式设置可采用附加的多行文本属性。对于多行文本而言，有三种操作模式。**setMultiLineTextMode** 方法可用于指定以下模式之一：

- MULTILINE_TEXT_OFF － 标注采用单行文本格式。
- MULTILINE_TEXT_ON － 采用标注文本中的现有换行，生成多行标注。

- MULTILINE_TEXT_COMPUTE － 标注文本将由标注引擎来动态评估，确定将标注文本格式设置为多行。此模式占有的系统运行时资源开销最大。

多行文本的默认行为是 off。

在第 246 页提供的代码示例介绍了多行文本标注的使用方法。

标注可视性

MapJ API 提供了若干种控制标注可视性的方式：设置缩放范围、是否允许文本重复或重叠以及设置标注优先级。

设置标注的缩放范围和设置图层的缩放类似。为此需要确定的是在哪一级别 （地图上的平面距离）显示标注，然后设置图层的 BaseLabelProperties 的最大值和最小值。

注：　这一缩放设置不适用于与标注专题相关的标注属性。

如果有两个图元采用了同一名称，可使用 **setDuplicationAllowed** 方法。该方法可用于实现同时标注两个图元。例如，现有一个名为纽约的州界和一个名为的纽约的市界。

借助于 **setOverlapAllowed** 方法可在聚焦区域中实现多个标注的可视性。重叠标注的默认行为是 False。在使用重叠标注时务必谨慎，标注过多反而会令地图难于辨识。

要控制某一区域内的标注密度，可设置每个图层的优先级。**setOverridePriority** 方法设置是使用默认的优先级还是使用覆盖值。覆盖值通过 **setPriority** 方法设置。默认情况下，如果 setDuplicationAllowed 或 setOverlapAllowed 设置为 True，图层列表中排位靠前的图层的标注在地图绘制时具有较高的优先级。

setPriority 方法可用于更改图层标注的优先级。默认的标注优先级值由以下等式给定：（图层数 － 图层位置）* 10。因此，如果 Layers 对象包含 20 个图层，第 5 位图层的默认标注优先级将为 150。如果图层所在的 Layers 对象增减其他图层，图层的默认标注优先级可能会随之发生变化。较高的值具有的优先级也高。如果出现重叠或重复，则将会渲染优先级较高的标注。

以下是设置影响标注可视性的若干 LabelProperties 方法的示例：

```
/* This example allows overlap, uses the second column to label,
increases the label priority for this layer to 200, and sets zoom
labeling from 10 to 30 kilometers.
*/
LabelProperties labelProp = new LabelProperties();
labelProp.setOverridePriority(true);
labelProp.setPriority(200);
labelProp.setLabelColumn(1);
labelProp.setOverlapAllowed(true);

// create a BaseLabelProperties
```

```
BaseLabelProperties baseLabel = new BaseLabelProperties(labelProp);
baseLabel.setZoom(true);
Distance maxDist = new Distance(30.0, LinearUnit.kilometer);
Distance minDist = new Distance(10.0, LinearUnit.kilometer);
baseLabel.setZoomMax(maxDist);
baseLabel.setZoomMin(minDist);
layer.setLabelProperties(baseLabel;
```

标注位置

LabelProperties 类提供了用于控制标注位置的多种方法，其中包括：

- 与标注点对齐方式
- 自标注点的偏移量
- 标注是否绕线图元旋转 （可选）
- 标注是否遵循线图元的路径 （可选）
- 在缩放变化时是否重新计算标注位置 （可选）

图元的标注位置使用以下算法计算：初始位置位于 Geometry 对象的 **getLabelPoint** 方法返回的标注点。若干该方法返回为空 （对于某些图元可能出现），与此相应的标注点将粗略计算为区域中心或直线的中点。初始位置随后将按照对齐和偏移来进行调整。

但是，如果设置了 “Follow Path” 属性，则标注位置将不从 getLabelPoint 计算，而是从路径动态进行计算。请参阅第 252 页的*沿特定路线标注* 。

对齐

标注可水平和垂直与标注点对齐。此处的对齐特指最靠近标注点的标注边框的边。左对齐表示标注边框的左边缘最靠近标注点 （标注显示在右侧）。

水平对齐可指定左、中或右对齐，如未指定，则使用默认对齐方式。默认情况下，VectorGeometries 为居中对齐， PointGeometries 为左对齐。

垂直对齐可以设置为基于基线的顶部、底部或居中。顶部垂直对齐标识边界框的顶边最靠近标注，以便标注显示在标注点之下。如为指定垂直对齐方式，则默认的行为是基线。

偏移量

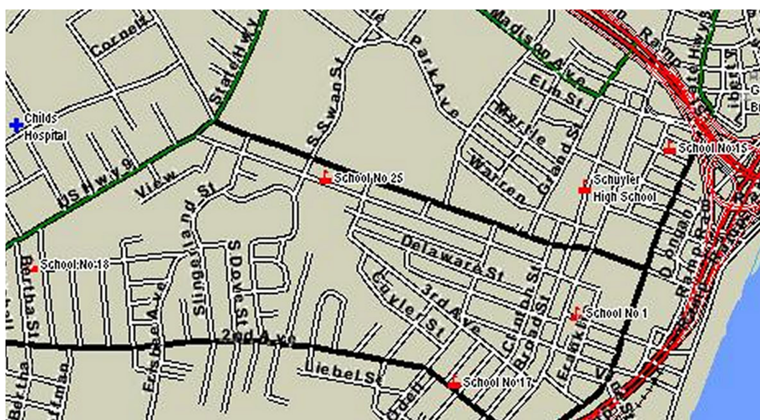
标注定位的第二个元素是偏移量的值。偏移量位置的值采用相对于用户空间的设备单位。请注意在 Java 的标准用户空间中， Y 轴正向位于 X 轴之下 （Y 的正值向下）。如未指定偏移量，则将使用默认值。区域的默认偏移量为 (0, 0)，直线的默认值参照线宽而定，具体为 (0, -w/2)，其中的 w 为线宽。点的默认值参照点的符号大小而定，具体为 (s/2 + 2, -s/2 + 2)，其中 s 是符号大小。

绕直线旋转标注

此外，还有一个标注位置元素控制标注是否遵循插图编号直线的斜率，默认为是。使用 **setLineLabelHorizontal** 方法即可覆盖该属性。

沿特定路线标注

标注可设置为在渲染时遵循图元几何形状的路线。这称为曲线标注，此操作在运行时计算路线，同时考虑其他标注属性设置，例如对齐方式、偏移量和样式。曲线标注尤其适用于折线，但是也可用于多边形。此时标注将遵循多边形的边界所确定的路线。



除了多行文本之外，直线标注的所有属性均可用于曲线标注。此外，诸如强制标注水平属性等不适用于曲线的属性将会被忽略。标注所遵循的路线始终是基于其几何形状来动态计算的并拟合为当前视图的。因此，几何输出节点始终默认为计算曲线标注。

如果标注文本比图元几何形状长，则将会截断部分标注。相应的标注只在字词之间的空白字符处截断。如果标注字符长度不足既定路线的 2/3，则将不绘制标注。这一点和 MapXtreme Java 在不久的将来即支持的基于 XML 的导出格式 － 缩放向量图像规范之中的文本布放规则要求相吻合。

此外，对取消标注还可应用多种创意效果。光晕、框格和下划线均为可用于曲线标注和直线标注。曲线标注不支持多行文本。

设置标注沿曲线绘制将影响到地图的绘制速度。曲线标准是在运行时计算的，任意附加的计算都会影响到渲染速度。因此，在设置标注属性时必须考虑众多因素。要确保曲线标注与其对应的直线标注不重叠，需要占用大量资源开支。如果标注在特定缩放级别密度较高并出现堆叠现象，并且此时不允许重叠，则整个地图渲染的性能均会降低。因此对较小的地图应该考虑使用曲线标注，以便在可视性和快速渲染时间之间获取最佳的折衷。

此外，为了支持曲线标注， MapJ API 中添加了 LabelProperties 类的两个新方法。使用方法 isLabelFollowingPath() 来确定是令标准贴合标注几何形状的路线，还是令标注置于标注的相对几何中心。返回为 True 表示标注将遵循标注的几何形状路线。

如果 bFollowPath 对曲线标注设置为 True，可使用 setLabelFollowingPath(boolean bFollowPath)。对于标注的默认行为是使用标注几何形状的中心作为放置点 （无曲线标注）。

代码示例

```
// Set LabelFollowingPath to true to enable label splining
boolean bFollowPath = true;
labelProps.setLabelFollowingPath(bFollowPath);
```

几何计算模式

setGeometryCalculationMode 方法指定标注在地图缩放更改时是否重新定位。当相应的值设置为 GEOMETRY_COMPUTED 时，标准将会重新定位。当放大地图，从较宽视图更改为较近的视图时，这一特性尤为实用。采用该特性时，标注将会重新调整到新位置，此时的计算以新拟合的地图视图为基础。默认的行为是 GEOMETRY_STATIC，此时在另一缩放级别上不重新计算位置。

以下示例展示了对齐方式、偏移量和几何计算模式属性的应用。

```
// change the horizontal and vertical alignments and offset and set the
// Geometry Calculation Mode to Compute.
LabelProperties labelProp = layer.getLabelProperties();
labelProp.setHorizontalAlignment(LabelProperties.HORIZ_AL IGN_RIGHT);
labelProp.setVerticalAlignment(LabelProperties.VERT_ALIGN _TOP);
labelProp.setOffset(new DoublePoint(10, -15));
labelProp.setGeometryCalculationMode(
    LabelProperties.GEOMETRY_COMPUTED);
BaseLabelProperties base_label = new BaseLabelProperties(labelProps);
layer.setLabelProperties(base_label);
```

合并标注属性

一个图层可以具有多个标注专题与其关联，这些相关的专题最终确定了图层上标注的外观。当标注专题排位较低时，其上的每个标注专题都可能对其部分相同属性作出更改。为了适应上述情况，并获取最终的预期标注结果，标注专题必须合并。

合并特指评估每个设置并确定标注专题在群组中的优先级的过程。每个标注专题均有优先级编号，该编号基于标注专题的群组中的编号以及其在群组中的排位。

合并时需要考虑以下两个注意事项：哪些设置优先，以及默认设置是表示其继续使用来自底层标注专题的设置，还是表示回退至标注专题的原始默认设置。

控制哪些设置优先可通过设置较高 **IntraGroupPriority** 编号来完成。此外，通过使用 **isIntraGroupPriorityCumulative** 方法，还可以控制优先级视为绝对还是相对值处理。

要确定默认值的含义，可考虑以下示例。最顶部的标注专题设置为默认对齐方式，但是下面的标注专题将其设置为 HORIZ_ALIGN_LEFT。此时默认值意味着什么？ 如果该属性设置为 NULL，则将会接收上一次的底层属性 (HORIZ_ALIGN_LEFT)。如果不为空，则将采用原始默认值，具体取决于所用的几何形状类型。

标注代码示例

本节提供了更改标注样式的示例。该代码还可参见于 MapXtreme Java 的 /codesamples 目录。

```
// set property to display labels
layer.setAutoLabel(true);

// Retrieve the LabelProperties from the layer and then assign the
// Rendition to our rend instance
LabelProperties labelProp = layer.getLabelProperties();
Rendition rend = labelProp.getRendition();

// Set the new rendition values
rend.setValue(Rendition.SYMBOL_FOREGROUND, Color.green);
rend.setValue(Rendition.SYMBOL_BACKGROUND, Color.blue);
rend.setValue(Rendition.FILTER_EFFECTS, Rendition.FilterEffects.HALO);
labelProp.setRendition(rend);
BaseLabelProperties base_label = new BaseLabelProperties(labelProps);
layer.setLabelProperties(base_label);

// Create an ImageRequestComposer
```

```
ImageRequestComposer imageRC =  ImageRequestComposer.create(
    myMap, 256, Color.blue, "image/gif");

// Create a MapXtremeImageRenderer
MapXtremeImageRenderer renderer = new
MapXtremeImageRenderer(mapxtremeServletURL);

// where mapxtremeServletUrl = URL to MapXtremeServlet,
// such as http://stockholm:8080/mapxtreme47/mapxtreme

// Render the map
renderer.render(imageRC);

// Render the map to the file:
renderer.toFile("comp.gif");
```

代码示例： OverrideLabelTheme

本代码示例展示如何覆盖图层标注。另请参阅 Javadocs 中有关 OverrideLabelTheme 的信息。

```
// obtain a reference to the target layer
FeatureLayer layer = null;

// name of the target layer whose labels' appearance will be altered by
// the OverrideLabelTheme
String TARGET_LAYER_NAME = "STATES";
if ((layer = (FeatureLayer)mapj.getLayers().getLayer(
    TARGET_LAYER_NAME)) != null)
{

    /* obtain the target layer's LabelThemeList, and add a new
    OverrideLabelTheme theme to it
    */
    LabelThemeList labelThemeList = null;
    if ((labelThemeList = layer.getLabelThemeList()) != null)
    {

        /* first, create a new LabelProperties object, which will hold
        the settings that define the characteristics of the new
        OverrideLabelTheme theme.
        */
        LabelProperties labelProperties = new LabelProperties();

        /* next, add a "LabelExpression" format string to our
        LabelProperties object, which describes how the text of the
        labels are to be fabricated. Specifically, the below format
        string forces label creation such that each label will contain
```

```
            field values from both the "state_name" and "state" columns of
            the underlying table.
            */
            String labelTemplate =
                "\"NAME:[\"+state_name+\"]\\nABBREV:[\"+state+\"]\"";
            labelProperties.setMultiLineTextMode(
                LabelProperties.MULTILINE_TEXT_ON);
            labelProperties.setLabelExpression(labelTemplate);

            /* add a new rendition object to our LabelProperties object,
            which will specify a new font to use when rendering all label
            text.
            */
            Rendition rend =new RenditionImpl();
            rend.setValue(Rendition.FONT_FAMILY,"Arial");
            rend.setValue(Rendition.FONT_WEIGHT,2);

            // NOTE: 2 == bold
            labelProperties.setRendition(rend);

            /* create a new OverrideLabelTheme object and add it to the
            target layer's LabelThemeList. Create a simple description for
            the OverrideLabelTheme
            */
            String THEME_DESCRIPTION_NAME =
                "test_for_override_label_theme";
            OverrideLabelTheme overrideLabelTheme = new
               OverrideLabelTheme(labelProperties,THEME_DESCRIPTION_NAME);
            labelThemeList.add(overrideLabelTheme);
        }
    }
```

代码示例： 范围 LabelTheme

以下代码示例展示如何创建范围标注专题。本示例也可参阅 Javadocs 中有关 RangedLabelTheme 的信息。

```
// obtain a reference to the target layer
FeatureLayer layer = null;

// name of the target layer whose labels' appearance will
// be altered by the RangedLabelTheme
String TARGET_LAYER_NAME = "STATES";
if ((layer = (FeatureLayer)mapj.getLayers().getLayer(
    TARGET_LAYER_NAME)) != null)
{

// obtain the target layer's LabelThemeList, and add a
```

```
// new RangedLabelTheme theme to it
LabelThemeList labelThemeList = null;
if ((labelThemeList = layer.getLabelThemeList()) != null)
{

    /* generate a column statistics object for a specific column within
    our target table name of a column in the target layer's underlying
    table, which the created  RangedLabelTheme will be based upon.
    */
    String TARGET_COLUMN_NAME = "POP_1990";
    ColumnStatistics columnStatistics =
    layer.fetchColumnStatistics(TARGET_COLUMN_NAME);

    /*
     generate a List of breakpoints, such that each breakpoints
    represents a specific numeric range into which the records of our
    table will be logically grouped.
    */

    //number of breakpoint ranges to create for the RangedLabelTheme
    int NUMBER_OF_BREAKPOINTS = 10;
    List breakPointSeries = (List) Bucketer.computeDistribution
        (NUMBER_OF_BREAKPOINTS,columnStatistics,
        Bucketer.DISTRIBUTION_TYPE_EQUAL_COUNT);

    /* create two(2) rendition objects that will act as the end-points
    for a range of renditions object that are to be created
    */
    Rendition rendStart = new RenditionImpl();
    rendStart.setValue(Rendition.FONT_FAMILY,"Arial");
    rendStart.setValue(Rendition.FONT_WEIGHT,2); // NOTE: 2 == bold
    rendStart.setValue(Rendition.FONT_SIZE,12);
    rendStart.setValue(Rendition.SYMBOL_FOREGROUND,Color.black);
    Rendition rendEnd = new RenditionImpl();
    rendEnd.setValue(Rendition.FONT_FAMILY,"Arial");
    rendEnd.setValue(Rendition.FONT_WEIGHT,2); // NOTE: 2 == bold
    rendEnd.setValue(Rendition.FONT_SIZE,24);
    rendEnd.setValue(Rendition.SYMBOL_FOREGROUND,Color.red);

    /* using the two(2) rendition objects, create a series of rendition
    objects that represent a gradation from the start rendition to the
    end rendition
    */
    List renditionSeries =(List) LinearRenditionSpreader.spread(
        NUMBER_OF_BREAKPOINTS,rendStart, rendEnd);

    /* create an ArrayList of LabelProperties objects, such that each
    LabelProperties object within this vector is assigned a rendition
    object from our above created series of rendition objects
```

```
        */
        ArrayList labelPropertiesSeries = new ArrayList();
        LabelProperties labelProperties = null;
        for (int i = 0; i < NUMBER_OF_BREAKPOINTS; i++)
        {
            labelProperties = new LabelProperties();
            labelProperties.setRendition(
                (Rendition)renditionSeries.get(i));
            labelProperties.setLabelColumn(TARGET_COLUMN_NAME);
            labelPropertiesSeries.add(labelProperties);
        }

        /* create a new RangedLabelTheme object and add it to ourconcerned
        layer's LabelThemeList object. Create a simple description for the
        RangedLabelTheme.
        */
        String THEME_DESCRIPTION_NAME = "test_for_ranged_label_theme";
        RangedLabelTheme rangedLabelTheme = new RangedLabelTheme(
            TARGET_COLUMN_NAME, // java.lang.String
            breakPointSeries, // java.util.List
            labelPropertiesSeries, // java.util.List
            THEME_DESCRIPTION_NAME // java.lang.String
            );
        labelThemeList.add(rangedLabelTheme);
    }
```

样式概览

Rendition 对象是一个显示属性的小集合，相应的属性控制地图上显示的图元或将要显示的标注的某一方面。在这个小集合中，用户只需设置相关的显示属性。其余的属性将借助于合并或包含其他样式。这同专题在 MapXtreme Java 中的工作机制相同。一个专题包含的一个样式只更改图元的一两个方面 （例如区域填充颜色），而不会更改其他属性 （例如区域边缘的颜色、宽度等）。

MapXtreme Java 支持众多显示属性，可充分利用 Java2D API 的所有渲染功能。其中包括用于直线和区域的符号图案、虚线和平行线、向量符号等等。

此外，改进的 RDBMS 数据源 （Oracle Spatial、 Informix、 SQL Server 等）还支持每图元（记录）样式。借助于这一特性，可以对源自 RDBM 数据源的图元的显示实现更加精确的控制，以便其在外观上看起来更加和 MapInfo 用户利用 TAB 文件所获取的地图相似。

本节介绍每图元样式和样式属性。

样式属性

样式属性用于说明地图图元的显示方式。样式 API 支持三种类别的属性：填充、单笔填充和符号。

填充属性控制区域的填充方式。单笔属性控制线条 （或线几何对象或区域的边缘）的绘制方式。符号属性控制如何绘制用于点几何对象、线标记或符号填充的符号。

填充属性

区域既可采用实体颜色填充，也可使用符号填充。要设置颜色，可指定表示所需颜色的 Java Color 对象。例如：

```
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.FILL, Color.red);
```

填充还可以通过使用填充图案的符号来定义。符号可以是支持的任意符号类型 （字体、图像或向量）。符号图案可以填充区域或较宽的直线 （可将较宽的直线视为在直径范围中使用 STROKE 画刷填充的多边形 STROKE_WIDTH 单位）。该符号用于创建重复绘制到该区域的 “平铺”。

对于平铺填充，可以作出如下理解。假设构成平铺面的每个块都具有一个图案，该图案与指定用于区域填充的符号样式相吻合。该区域就像在平铺面上的一张纸上穿个孔。平铺图案将显示穿过这个孔。如果有多个区域使用相同的符号图案，则符号块将匹配或列成一队，以便其可以显示为穿透同时绘制的所有图元。

以下是指定沼泽区域的符号图案示例。此时将创建一个 swamp 符号 （如表示沼泽草地的 GIF 文件）。这一 'swamp grass' 符号将用于平铺填充该区域，得到的外观显示为满是沼泽草地的区域。

```
Rendition rendSymbol = new RenditionImpl();
rendSymbol.setValue(Rendition.SYMBOL_MODE,
Rendition.SymbolMode.IMAGE);
rendSymbol.setValue(
Rendition.SYMBOL_URL, "http://www.myhost.com/image/swamp.gif");
Rendition rendFill = new RenditionImpl();
rendFill.setValue(Rendition.FILL, rendSymbol);
```

对于实体图案和符号图案，还可以使用 Rendition.FILL_OPACITY 属性来控制画刷的不透明性。值的范围介于完全透明填充的 0.0 到完全不透明填充的 1.0 之间。

```
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.FILL, Color.blue);
rend.setValue(Rendition.FILL_OPACITY, 0.5f);
```

渐变

MapXtreme Java 现在可以使用线性或半径颜色渐变填充方式来填充多边形。

线性渐变

线性渐变沿直线通过一系列颜色过渡。颜色端点用于指定在特定位置所需的颜色。线性渐变所遵循的直线称为渐变线。默认渐变线的方向为从左至右。这一行为可通过构建用于该渐变线的带有特定起止点的 LinearGradient 来取代。用于该渐变线的所有坐标均使用 0 （表示 0%）和 1. 0 （表示 100%）之间的数字指定为所要填充的对象的长度和宽度的百分比。这意味着从右向左指定渐变线，用户将指定起点 (1.0, 0.0) 和终点 (0.0, 0.0)。此外，还存在的其他可能性有对角渐变线，以及完全在所要填充或单笔填充的对象之中起止的渐变线。此行为在缩放向量图形 (SVG) 中线性渐变之后建模。如果要着色的点位于渐变线外，则扩展方法将指定用于点的着色的规则。

以下示例创建自区域左侧开始并于区域右侧终止的 LinearGradientat。颜色渐变本身在沿向量 1/4 处开始，在沿向量 3/4 处终止。

```
// create two color stops: one a quarter of the way "down"
// the vector and the other three-quarters of the way "down" the
// vector.
List colorStops = new ArrayList();
```

```
colorStops.add(new ColorStop(Color.RED.brighter(), 0.25));
colorStops.add(new ColorStop(Color.RED.darker(), 0.75));

// the gradient vector travels from left to right through the middle of
// the region
DoublePoint startPoint = new DoublePoint(0.0, 0.5);
DoublePoint endPoint = new DoublePoint(1, 0.5);

// create LinearGradient
LinearGradient linearGradient = new LinearGradient(
    colorStops, SpreadMethod.PAD, startPoint, endPoint);

// set the fill to be the LinearGradient
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.FILL, linearGradient);
```

半径颜色渐变

半径颜色渐变沿圆通过一系列颜色过渡。如果所要填充或单笔填充的对象边界框不是方形，则渐变路径将为椭圆以匹配边界框的纵横比。半径颜色渐变的限制是由圆确定的，中心为 0% 停止点，外部轮廓是 100% 的停止点。外圈是通过中心和半径指定的。中线和半径是使用 0 和 1 之间的值指定为所要单笔填充或填充的边界矩形的宽度和高度的百分比。中心和半径的默认值分别为 (0.5, 0.5) 和 0.5。这将圆置于中部，其半径为对象宽度和高度的一半。如果要填充的对象边界框不是方形，则圆将成为具有两个半径的椭圆。一个半径是宽度的百分比，另一个半径是高度百分比。颜色端点用于指定在特定位置所需的颜色。如果要着色的点位于圆外，则扩展方法将指定用于点的着色的规则。

以下示例创建始自于对象中心和渐变为所要填充对象一半的半径的 RadialGradient。

```
// create two color stops: one a quarter of the way "down"
// the vector and the other three-quarters of the way "down" the
//vector.
List colorStops = new ArrayList();
colorStops.add(new ColorStop(Color.RED.brighter(), 0.25));
colorStops.add(new ColorStop(Color.RED.darker(), 0.75));

// the gradient vector travels from left to right through the middle of
// the region
DoublePoint center = new DoublePoint(0.5, 0.5);
double radius = 0.5;

// create RadialGradient
RadialGradient radialGradient = new RadialGradient(
    colorStops, SpreadMethod.PAD, center, radius);
```

```
// set the fill to be the RadialGradient
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.FILL, radialGradient);
```

单笔填充属性

单笔填充属性控制如何显示直线或区域的边。这些属性控制单笔填充图案、线宽、线段连接点和端点、虚线模式和更多特性。

Rendition.STROKE 属性控制在绘制直线或边时使用的图案。属性值可是是实体颜色、用于指定符号图案的样式或渐变。这与此前所述的 Rendition.FILL 属性非常相似。当值为类似 FILL 属性的样式时，样式的符号属性将用于创建要用于对直线进行片断填充的符号。通常，直线上的符号填充只有在 STROKE_WIDTH > 1 时有意义。

Rendition.STROKE_WIDTH 控制直线宽度 （单位为磅）和 Rendition.STROKE_OPACITY、不透明性 （0.0 为透明， 1.0 为不透明）。

```
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.STROKE, Color.red);
rend.setValue(Rendition.STROKE_WIDTH, 3.2f);
rend.setValue(Rendition.STROKE_OPACITY, 0.3f);
```

平行线

渲染引擎支持渲染和基线平行的一条或多条直线。Rendition.STROKE_PARALLELARRAY 属性可以包含一个或多个 Rendition.ParallelLine 对象的数组。每个 Rendition.ParallelLine 对象均包含要绘制的与基线平行的直线的偏移量和样式。

偏移量为渲染引擎确定了距离基线多少个单位来绘制直线。目前所有单位均以打印机点（1/72 英寸）指定。偏移量可以为任何数字，其中 0 表示在基线上绘制平行线 （无偏移），+N 表示在基线右侧 N 个单位绘制平行线， -N 表示在基线左侧 N 个单位绘制平行线。左右根据基线方向判断。如果直线的第一个点起自屏幕左侧，而下一个点位于该点右侧，则 +N 偏移量将令平行线位于下方或基线行程方向的右侧。

每个平行线都具有单独的样式来指定其绘制方式。

平行线始终在基线渲染之后绘制。

铁路线是平行线的典型示例。这相当于有一条透明基线和两条平行线，每侧一条，两者与基线等距。

```
Rendition rendParallel = new RenditionImpl();
rendParallel.setValue(Rendition.STROKE, Color.black);
Rendition.ParallelLine parallel1 = new Rendition.ParallelLine(
    3.0f, rendParallel);
Rendition.ParallelLine parallel2 = new Rendition.ParallelLine(
    -3.0f, rendParallel);
```

```
Rendition.ParallelLine[] parallelArray = {parallel1, parallel2};
Rendition rendBaseLine = new RenditionImpl();
rendBaseLine.setValue(Rendition.STROKE_OPACITY, 0f);
rendBaseLine.setValue(Rendition.STROKE_PARALLELARRAY, parallelArray);
```

虚线

虚线由数字 (float[]) 数组定义并指定为数字对。每个数字对均指定短线的长度以及其间距。例如，值为 [5,3] 的虚线将采用 5 个单位的短线和 3 的单位间距。STROKE_DASHARRAY 属性可以和任意类型直线或边一起使用。

STROKE_DASHOFFSET 属性控制在虚线数组中采用多少个单位来开始虚线图案。例如，假定 STROKE_DASHARRAY 属性值为 [5, 3]。如果没有设置 STROKE_DASHOFFSET 或者设置为 0，则虚线图案将采用 5 个单位的短线和 3 个单位的间距。如果 STROKE_DASHOFFSET 设置为 3，则该线将采用 2 个单位的短线和 3 个单位的间距，然后是 5 个单位的短线和 3 个单位的间距。

显示在建或规划地下电缆是虚线的示例。

```
// Create a line that draws two points, skips four points, and so on
dashArray = new float[] {
    2, 4
};
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.STROKE_DASHARRAY, dashArray);

// Create a dashed line that draws two points, skips four,
// ... and so on except start drawing 4 points into the
// line. This has the effect of starting the line with two
// empty points, two drawn, four empty, two drawn, etc.
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.STROKE_DASHARRAY, dashArray);
rend.setValue(Rendition.STROKE_DASHOFFSET, 4);
```

直线标记

除了符号旋转以匹配直线角度之外，单笔填充标记和符号图案类似。使用直线标记标注带有重复符号的直线路径。

例如，使用 Rendition.STROKE_MARKERARRAY 和小汽车图像来渲染交通高峰时公路上一个挨一个的小汽车。此时需要将 Rendition.SYMBOL_MODE 设置为 Rendition.SymbolMode.IMAGE，将 Rendition.SYMBOL_URL 指向透明小汽车图像的栅格文件的 URL。（符号说明如下）。随后将直线几何对象的 Rendition.STROKE_MARKERARRAY 设置为符号样式。在渲染时，直线几何对象将基于该对象的其他属性绘制，随后小汽车图像将会旋转并沿每条线段的路径重复绘制（线段是介于直线几何对象的任意两点之间的路径）。

```
Rendition rendSymbol = new RenditionImpl();
rendSymbol.setValue(Rendition.SYMBOL_MODE,
    Rendition.SymbolMode.IMAGE);
rendSymbol.setValue(Rendition.SYMBOL_URL,
    "http://www.myhost.com/image/car.gif");
Rendition.Marker marker = new Rendition.Marker(rendSymbol);
Rendition rendLine = new RenditionImpl();
rendLine.setValue(Rendition.STROKE_MARKERARRAY,
    new Rendition.Marker[]{marker});
```

直线端点，连接点

样式 API 提供了多种方式来完成直线的端点和将直线联接在一起。

使用带有 STROKE_LINECAP 属性的 Rendition.LineCap 来完成带有原型或方形端点装饰或没有装饰的直线 （分别使用圆、方或端点属性）。

与此类似， Rendition.STROKE_LINEJOIN 属性有三种连接线段的方式：连接外角 （斜角）、延伸外边连接 （斜接）和圆角 （圆）。

渐变单笔填充

MapXtreme Java 现在可使用线性或半径颜色渐变单笔填充。有关线性和半径颜色渐变的完全说明，请参阅第 260 页的*渐变* 。

以下示例创建自区域左侧开始并于区域右侧终止的 LinearGradientat。颜色渐变本身在沿向量 1/4 处开始，在沿向量 3/4 处终止。

```
// create two color stops: one a quarter of the way "down"
// the vector and the other three-quarters of the way "down" the
//vector.
List colorStops = new ArrayList();
colorStops.add(new ColorStop(Color.RED.brighter(), 0.25));
colorStops.add(new ColorStop(Color.RED.darker(), 0.75));

// the gradient vector travels from left to right through the middle of
// the region
DoublePoint startPoint = new DoublePoint(0.0, 0.5);
DoublePoint endPoint = new DoublePoint(1, 0.5);

// create LinearGradient
LinearGradient linearGradient = new LinearGradient(
    colorStops, SpreadMethod.PAD, startPoint, endPoint);

// set the fill to be the LinearGradient
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.STROKE, linearGradient);
```

以下示例创建自区域左侧起始并于区域右侧终止的 LinearGradient。颜色渐变本身在沿向量 1/4 处开始，在沿向量 3/4 处终止。

```
// create two color stops: one a quarter of the way "down"
// the vector and the other three-quarters of the way "down" the
// vector.
List colorStops = new ArrayList();
colorStops.add(new ColorStop(Color.RED.brighter(), 0.25));
colorStops.add(new ColorStop(Color.RED.darker(), 0.75));

// the gradient vector travels from left to right through the middle of
// the region
DoublePoint startPoint = new DoublePoint(0.0, 0.5);
DoublePoint endPoint = new DoublePoint(1, 0.5);

// create LinearGradient
LinearGradient linearGradient = new LinearGradient(
    colorStops, SpreadMethod.PAD, startPoint, endPoint);

// set the fill to be the LinearGradient
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.STROKE, linearGradient);
```

符号属性： 字体、图像和向量

除了标记点位置之外，MapXtreme Java 中的符号还可以执行众多操作。如上所述，符号可以用作样式来填充区域、较宽的直线或直线标记。符号可分为三种类型：字体、图像和向量。

字体符号

诸如 Type 1 或 TrueType 之类 Java 2D 平台支持的任意字体均可用作符号。MapXtreme 提供了在地图绘制中的常用众多 TrueType 符号集，其中包括：

- MapInfo Cartographic
- MapInfo Transportation
- MapInfo Real Estate
- MapInfo Miscellaneous
- MapInfo Oil & Gas
- MapInfo Weather
- MapInfo Arrows
- MapInfo Shields
- MapInfo Symbols
- 地图符号

在安装之后，这些字体均位于 /<OPERATING_SYSTEM>/fonts 目录之下。如果使用的不是 Windows 平台，还必须将这些字体注册到操作系统，以便可在 MapXtreme Java 版中使用相应字体。要通过程序访问这些字体，可直接使用其一般名称 （例如 MapInfo Arrows、MapInfo Cartographic 和 MapInfo Symbols）。有关字体的预览，可参阅第 374 页的 *MapXtreme Java TrueType 符号* 。

要查看字体，可使用操作系统的字体查看工具，例如用于 Windows 的 Unicode 字符映射。

将 Rendition.SYMBOL_MODE 属性设置为 Rendition.SymbolMode.FONT，字体属性 （如 Rendition.FONT_FAMILY）与 Rendition.SYMBOL_STRING 属性一起指定字体符号。使用字体符号时，可选择字体系列、大小、背景色和前景色，以及类似粗体、斜体、下划线、光晕、框格和轮廓线的创意效果。

使用 FONT_SIZE，可借助于表示磅值的数字或表示实际物理高度的距离对象指定大小。本示例显示的是 12 磅的符号。

```
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.SYMBOL_MODE, Rendition.SymbolMode.FONT);
rend.setValue(Rendition.FONT_FAMILY, "MapInfo Cartographic");
rend.setValue(Rendition.FONT_SIZE, 12);
rend.setValue(Rendition.SYMBOL_STRING, String.valueOf((char)33));
```

要将字体符号指定为 12 英里的高度，以便该符号可在不同的缩放级别适当缩放，可参考以下示例：

```
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.SYMBOL_MODE, Rendition.SymbolMode.FONT);
rend.setValue(Rendition.FONT_FAMILY, "MapInfo Cartographic");
Distance fontsize = new Distance(12, LinearUnit.mile);
rend.setValue(Rendition.FONT_SIZE, fontsize);
rend.setValue(Rendition.SYMBOL_STRING, String.valueOf((char)33));
```

图像符号

符号也可通过公司徽标之类的图像 （GIF、 JPEG、 PNG 等）来表示。当 Rendition.SYMBOL_MODE 属性设置为 Rendition.SymbolMode.IMAGE 时，Rendition.SYMBOL_URL 属性将用于从指定 URL 检索图像。Rendition.SYMBOL_URL 属性包含一个 URL （如 http://*myhost*.com/image/truck.gif）。

例如，再次使用此前重复透明的小汽车图像的示例。该示例使用 Rendition.STROKE_MARKERARRAY 来指定小汽车符号沿直线重复。

```
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.SYMBOL_MODE, Rendition.SymbolMode.IMAGE);
rend.setValue(Rendition.SYMBOL_URL,
    "http://myhost.com/image/car.gif");
```

SYMBOL_URL 可以是包含数据的任意符合 RFC2397 (**http://www.faqs.org/rfcs/rfc2397.html**) 规范中指定标准的 URL。数据 URL 允许数据是 URL 本身的组成部分。借助于此， MapXtreme Java 地图数据本身可以包含更多信息，而不只是依赖于服务器或特定目录结构的存在与否。

MapXtreme Java 提供了可用于标记地图的定制符号 GIF 图像集。在安装之后，该图像集可见于 /server/symbols/custom 目录。有关符号的缩略图，请参阅*附录 F: 定制符号* 。

用于动画图像的 OVERLAY_IMAGE

图像符号也可表示为动画图像。按照 MapXtreme Java 术语，动画图像称为覆盖图像。覆盖图像是样式对象的属性。当 Rendition.SymbolMode 属性设置为 Rendition.SymbolMode.**OVERLAY_IMAGE** 时， Rendition.SYMBOL_URL 属性将用于从指定 URL 检索图像。 （对于生成任意类型的图像符号而言均为同样的行为。）此属性只适用于点图元。

覆盖图像需要使用名为 **EncodedImageRenderer** 的特殊渲染器，该渲染器采用新的 MIME 类型，即 application/encodedimage+xml;image/xxx，其中 xxx 可为 gif、 jpg、 png 等后缀。为表明所需的动画图像信息， Enterprise XML 协议中的 MapImageRequest 返回了一个图像（image/gif、 image/jpeg 等）和一个 **MapImageResponse**。MapImageResponse 是一个包含基础地图的 XML 文档和一个点覆盖的列表。每个点覆盖元素包含的信息说明其样式和相对于基础地图的位置。

有关 EncodedImageRenderer 的详细信息，请参阅*第 11 章：渲染涉及的考虑因素* 。有关 MapImageRequest 的详细信息，请参阅*第 16 章：XML 协议* 。

向量符号

如果字体和图像符号没有提供所需的符号表示，可以自行绘制所需象征符号。MapXtreme Java 支持可由 Java2D Shape 接口指定的任意形状的向量符号。

当 Rendition.SYMBOL_MODE 属性设置为 Rendition.SymbolMode.SHAPE 时， Rendition.SYMBOL_SHAPE 属性将用于创建符号。Rendition.SYMBOL_SHAPE 属性包含一个 Rendition.SymbolShape 对象。此对象由 Rendition 对象和实施 java.awt.Shape 接口的对象构成。

例如，java.awt.geom 程序包中的众多对象（如 Rectangle2D、多边形等）实施 Shape 接口。此外，还可以使用 java.awt.geom.GeneralPath 对象，并借助于 moveTo、 lineTo 等命令来指定更复杂的几何对象。在显示 Shape 接口时，将使用 Rendition.SymbolShape 的 Rendition 对象。即 Rendition 可指定用于区域形状的 FILL 图案。

```
Rendition rendShape = new RenditionImpl();
rendShape.setValue(Rendition.FILL, Color.red);
Rectangle2D rect = new Rectangle2D.Float(0, 0, 10, 10);
```

```
Rendition.SymbolShape symbolShape = new Rendition.SymbolShape(
    shape, rend);
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.SYMBOL_MODE, Rendition.SymbolMode.SHAPE);
rend.setValue(Rendition.SYMBOL_SHAPE, symbolShape);
```

调整符号大小

MapXtreme Java 支持调整以实际物理距离单位指定的绝对大小。您可能需要为符号设置绝对大小，以便地图在不同缩放级别重画时，符号以正常的比例显示。此前只能使用相对大小，即符号使用仿射转换 (SYMBOL_TRANSFORM) 放大为其高度的 2 倍。在缩放级别变化时，这样做并非始终恰当。

符号调整大小是通过 Rendition 类的新属性完成的，该类可用于以高度和宽度指定符号的实际物理大小，例如 2 x 4 英里。符号 SYMBOL_SIZE 属性是以下类型的符号固有属性：图像符号、覆盖图像符号和形状 （向量）符号。字体符号继续使用 FONT_SIZE 用于缩放。

SYMBOL_SIZE 属性将采用 Size 对象，该对象定义符号的高度和宽度以及尺寸的距离单位。

此外还可组合使用符号的相对和绝对大小调整。此时， SYMBOL_SIZE 将与 SYMBOL_TRANSFORM 一起组合使用。例如，如果 SYMBOL_SIZE 设置为 2 x 3 英里，并且 SYMBOL_TRANSFORM 设置乘以 2，此时该符号的绘制尺寸为 4 x 6 英里。

使用 SYMBOL_SIZE 集绘制地图的速度要比不使用该属性绘制符号稍慢。实际物理尺寸的高度宽度需要转换为屏幕像素，但其对于地图绘制速度的影响可以忽略不计。

有关详细信息，请参阅 Javadocs 中的 Rendition 和 Size 类。

代码示例： 调整符号大小

```
// The symbol size can be in more than one linear unit
Distance symbolWidth = new Distance(50, LinearUnit.kilometer);
Distance symbolHeight = new Distance(75, LinearUnit.mile);
Size symbolSize = new Size(symbolWidth, symbolHeight);
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.SYMBOL_MODE, Rendition.SymbolMode.IMAGE);
rend.setValue(Rendition.SYMBOL_URL,
    "http://host:8080/locationOfImage/image.gif");
rend.setValue(Rendition.SYMBOL_SIZE, symbolSize);

// Transforms are applied after scaling the image the specified size.
So this scales the symbol to 100 kilometers by 225 miles.
rend.setValue(Rendition.SYMBOL_TRANSFORM,
AffineTransform.getScaleInstance(2, 3));
// now add the rendition to a theme, save to a database, etc....
```

代码示例： 创建用于直线标记的向量符号

本示例代码创建了构成 X 和菱形的标记数组。这意味着在此直线的顶部将有一串标记：X 后跟一个菱形，直至达到此直线的末端。

此代码也可见于 Javadocs。

```
// create a shape to represent the first marker (an X)
java.awt.geom.GeneralPath path = new java.awt.geom.GeneralPath();
path.moveTo(-3, -3);
path.lineTo(3, 3);
path.moveTo(3, -3);
path.lineTo(-3, 3);

// each marker is actually a symbol
Rendition symbolRend =  new RenditionImpl();
symbolRend.setValue(Rendition.STROKE, Color.green);
Rendition markerRend =  new RenditionImpl();
symbolShape = new Rendition.SymbolShape(path, markerRend);

// each marker gets its own rendition
Rendition marker1Rend = new RenditionImpl();
marker1Rend.setValue(Rendition.SYMBOL_MODE,
Rendition.SymbolMode.SHAPE);
marker1Rend.setValue(Rendition.SYMBOL_SHAPE, symbolShape);

// create a shape to represent the second marker (an diamond)
GeneralPath path = new GeneralPath();
path.moveTo(3, 0);
path.lineTo(6, -3);
path.moveTo(9, 0);
path.lineTo(6, 3);
path.closePath();

// each marker is actually a symbol
Rendition symbolRend =new RenditionImpl();
symbolRend.setValue(Rendition.STROKE, Color.green);
symbolShape = new Rendition.SymbolShape(path, markerRend);

// each marker gets its own rendition
marker2Rend = new RenditionImpl();
marker2Rend.setValue(Rendition.SYMBOL_MODE,
Rendition.SymbolMode.SHAPE);
marker2Rend.setValue(Rendition.SYMBOL_SHAPE, symbolShape);
Rendition.Marker[] markerArray = {
    new Rendition.Marker(marker1Rend),
    new Rendition.Marker(marker2Rend)
};
```

```
// the rendition with which to draw
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.STROKE_MARKERARRAY, markerArray);
```

代码示例： 字体属性

以下代码示例展示如何设置样式的 FONT_XXX 属性

```
// Use the MapInfo Miscellaneous font
rend.setValue(Rendition.FONT_FAMILY, "MapInfo Miscellaneous");

// Make the font 14 pts
rend.setValue(Rendition.FONT_SIZE, 14);

// The weight should be greater than 0. 1.0 = normal, 2.0 = bold, etc.
rend.setValue(Rendition.FONT_WEIGHT, new Float(2));

// Make it a leaning garbage can
rend.setValue(Rendition.FONT_STYLE, Rendition.FontStyle.ITALIC);

// indicate that we are using a font for the symbol
rend.setValue(Rendition.SYMBOL_MODE, Rendition.SymbolMode.FONT);

// the number 4 is the garbage can
rend.setValue(Rendition.SYMBOL_STRING, "4");

// make the symbol yellow on a red background
rend.setValue(Rendition.SYMBOL_FOREGROUND, Color.yellow);
rend.setValue(Rendition.SYMBOL_BACKGROUND, Color.red);
```

代码示例： 填充属性

```
/* Create a rendition that is filled blue but allows 60 percent  of the
background to be visible (40 % transparent). */
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.FILL, Color.blue);
rend.setValue(Rendition.FILL_OPACITY, new Float(0.40));
```

代码示例： 使用符号绘制线条

```
// define a red line 7.5 pts wide
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.STROKE, Color.red);
rend.setValue(Rendition.STROKE_WIDTH, new Float(7.5f));

// demonstrate creating a rendition to draw a line with a symbol
Rendition symbolRend = new RenditionImpl();
symbolRend.setValue(Rendition.SYMBOL_MODE, Rendition.SymbolMode.FONT);
```

```
symbolRend.setValue(Rendition.FONT_FAMILY, "MapInfo Symbols");
symbolRend.setValue(Rendition.FONT_SIZE, 8);
symbolRend.setValue(Rendition.SYMBOL_STRING, "#");

// create rendition to draw lines using symbols
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.STROKE, symbolRend);
rend.setValue(Rendition.STROKE_WIDTH, new Float(5));
```

代码示例： 虚线和平行线

本示例展示 STROKE、 STROKE_DASHARRAY、 STROKE_PARALLEL ARRAY、 Layers 和 Layer。所生成的线条为红色的虚线中线，在距其 12 磅处，有一条白线和一条紫线与中线平行。

```
// rendition for the first parallel line
Rendition rend1 = new RenditionImpl();
rend1.setValue(Rendition.STROKE, new Color(1.0f, 0.0f, 1.0f));

// purple line rendition for second parallel line
Rendition rend2 = new RenditionImpl();
rend2.setValue(Rendition.STROKE, Color.white);
rend2.setValue(Rendition.STROKE_DASHARRAY, new float[] { 4, 4 } );

// Create array of parallel lines for the central line. The red and
// white dashed line is created by specifying a 0.0 for an offset.
Rendition.ParallalLine[] parallelLines = new Rendition.ParallelLine[]
{
    new Rendition.ParallelLine(12.0f,rend1);
    new Rendition.ParallelLine(0.0f,rend2);
};

// create rendition for Australia's borders.
Rendition borderRend = new RenditionImpl();
borderRend.setValue(Rendition.STROKE, Color.red);
borderRend.setValue(Rendition.STROKE_PARALLELARRAY, parallelLines);
```

命名样式

命名样式是一种命名资源，并且其表示的样式具有唯一的名称。NamedRendition 位于 com.mapinfo.graphics 文件包之内。

命名样式和 MapXtreme Java 中的其他命名资源具有相同的优点：

- 该资源以其名称所知，而非其属性。
- 该资源位于一点，但可从多个位置引用。
- 要更改应用程序或数据的外观或行为，只需更改资源，而非每个应用程序或数据文件。

正如所有其他命名资源一样，命名样式使用 Java 命名和目录接口 (JNDI) 应用程序编程接口 (API)。NamedRendition 对象透明处理所有和 JNDI 的交互。

创建命名样式

创建命名样式非常轻松：

```
NamedRendition namedRendition = new NamedRendition(providerUrl, name);
```

或

```
NamedRendition namedRendition = new NamedRendition(jndiContext, name);
```

其中：

providerUrl （字符串）是指向存储命名资源 （JDNI 上下文中的库）的对象的 URL

name （字符串）是和样式关联的名称

jndiContext （上下文）是此前通过 JNDI API 直接创建的 Context 对象。

命名样式是在其访问时才解析的。

如非属性值改变，命名样式将始终保持其名称，在属性值改变之后命名样式将只是一个样式。

命名样式实施 Rendition 接口，以便其可用于任意其他样式可用的场所。

MapXtreme Java 内置的预定义画笔、画刷和符号样式均可视为采用其自带的 brush_008、pen_057 等名称的 “命名资源”。这些样式均位于 mapxtreme47/resources/mistyles 目录之中。这些样式可用作命名样式的起点，用户可以基于此来进行自定义，并采用更加简明易懂的名称来进行保存相应的样式。MapXtreme Java 管理器的命名资源面板提供了一个管理这些样式的 GUI。请参阅*第 5 章：管理 MapXtreme Java* 。

使用 NamedResource 存储命名样式

任意样式均可通过 JNDI API 存储为命名样式。JNDI 上下文 (javax.naming.Context) 使用两种方法来将命名资源保存到命名资源库。如下所示：

- **bind**(String name, Object obj)
- **rebind**(String name, Object obj)

使用 bind() 方法可将全新的资源 （以前不存在）保存到库中。使用 rebind() 方法可更新库中已经存在的资源。

显而易见，所需的第一个对象是 JNDI 上下文。这既可以是 InitialContext (javax.naming.InitialContext)，也可以是 InitialContext （或通过 lookup() 经由 InitialContext）的子上下文。有关上下文和 InitialContexts 的详细信息，请参阅 Javadoc 中有关 JNDI API 的介绍。

要创建初始上下文，需要知道提供方的 URL，该 URL 是 （最可能是） NamedResourceServlet 的 URL。然后调用 NamedResourceContextFactory 类的 createInitialContext(providerURL) 工厂方法，如下所示：

```
Context initCtx =NamedResourceContextFactory.createInitialContext(
    "http://torpedo:8080/mapxtreme47/namedresource:);
```

现在只需要决定在何处存储命名样式，该位置是相对于命名资源库根的位置。切记 NamedResourceServlet 已知命名资源库的根，这一信息是通过 NamedResourceServlet 的初始参数 resourcesdir 指定的。

在名为 "my renditions" 的库的根下创建目录之后，即可在 "blue star" 子目录中保存特定样式。这一操作如下所示：

```
// blueStar Rendition was previously initialized with the
// desired symbol properties
// create a named resource out of the blueStar Rendition
NamedResource resource = new NamedResource(blueStar);

// Now save it via the container we obtained above
initCtx.bind("my renditions/blue star", resource);
```

在上例中， "myrenditions/blue star" 表示一个复合名称。在指定复合名称时，每个组件名都必须以 / （正斜线）间隔。

注： 命名资源 （样式）必须始终存储在库的根的子目录中。这些资源绝对不能直接存储在根中。

检索命名样式

要从命名资源库检索命名样式，可使用 com.mapinfo.graphics.NamedRendition 类。要检索存储在命名资源库的根的子目录 "my renditions" 之中的命名样式 "blue star"，只需创建一个新的 NamedRendition，并提供如下所示的提供方 URL 和资源名称 （相对于库的根）即可：

```
NamedRendition rendition = new NamedRendition(
    "http://torpedo:8080/mapxtreme47/namedresource",
    "my renditions/blue star");
```

NamedRendition 实施 Rendition 接口，因此可用于允许使用样式的任意场所。

每图元样式

MapXtreme Java 可在 MAPINFO_MAPCATALOG 中指定表级别的样式 （由 FeatureSet's getRendition 方法返回）。表级别的样式可用作相应表的所有图元的基础显示属性集。

在要用作每图元样式 （由图元的 getRendition 方法返回）的空间表中，可以指定一个附加列。这一每图元样式将与表 (FeatureSet) 的样式合并 （覆盖），以确定用于图元显示的属性集。如果没有每图元样式，则将会使用表级别的样式。

该样式列既可确定为在图层创建时 （通过编程或通过 LayerControl Bean）使用的参数，也可确定为源自 MAPINFO_MAPCATALOG 为的参数。空间表中的这一样式列可为 Null、MapBasic 样式字符串或 MapXtreme Java 样式。有关 MAPCATALOG 的详细信息，请参阅*附录 C: MAPINFO.MAPINFO_ MAPCATALOG* 。

代码示例： 每图元样式

```
/* Programatically create the rendition column at the layer creation
 * time.
 * Constructs the objects given the table name and information
 * describing the Rendition to use for each Feature
 * Parameters:
 *    tableName - The file name of the table.
 *    renditionColumn - The column containing Rendition information
 * for each Feature.
 *    renditionType - Object identifying the type of rendition data
 * found in the rendition column.
 */

//create the TABTableDescHelper
```

```
TABTableDescHelper tabTDHelper =  new TABTableDescHelper(
    new File("States_Rend.tab").getName(),
    "FeatureRendition", RenditionType.mapxtreme );

//create the TABDataProviderHelper
TABDataProviderHelper tabDPHelper = new
    TABDataProviderHelper("D:\\maps");

//create the LocalDataProviderRef
LocalDataProviderRef localhelper = new
    LocalDataProviderRef(tabDPHelper);

//add it to the layer
m_map.getLayers().addLayer(localhelper,tabTDHelper, "tabLayer");
```

15

专题地图绘制和分析

本章介绍 MapXtreme Java 中的专题地图绘制。专题地图绘制功能强大，可以实现对数据的分析和可视化。

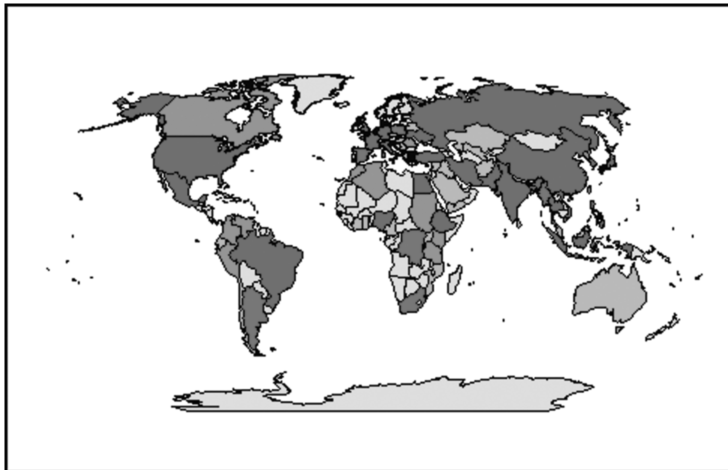
本章内容：

专题地图绘制概览

专题地图绘制特指根据特定专题通过影线表示地图的过程。专题通常是数据的片断。绘制地图时将使用图层数据，按照特定专题来采用影线表示。气候分布图就是最常见的专题地图示例之一。地图上的红色表示热 （温度高），蓝色表示冷 （温度低）。

专题通过颜色、填充图案或符号的影线来表示数据。使用专题地图显示数据可采用多种形式。通过根据数据中的特定值为地图对象指定颜色、图案或符号，即可创建不同的专题地图。

专题地图绘制实现了数据的可视化，并可突出显示数据趋势，这些信息是通过表格数据难以发现的。使用 ThemeList 中的方法和 Theme 接口，可以创建并定制专题影线。



专题类型

MapXtreme Java 版提供以下 4 种专题地图：

- **OverrideTheme** － 更改整个图层的样式
- **RangedTheme** － 划分数据范围，并根据范围值来确定影线表示
- **IndividualValueTheme** － 通过影线表示共享特定属性值的图元组
- **SelectionTheme** － 对用户定义的选定图元列表应用样式

具有专题的所有图层均使用 ThemeList 集合、Theme 接口和 Rendition 对象。每个特定的专题类型均可使用附加的对象。本章的后续内容对此作出了详细说明。

专题可以通过程序创建；如果是 RangedTheme 和 IndividualValueTheme，则可通过 AddTheme Wizard Bean 创建。专题可以通过图层控制来显示和控制。

专题图例

MapXtreme Java 可以为范围专题和个别值专题生成图例。图例中的信息直接绑定到相应专题。专题更改，图例亦随之更新。

AddTheme 向导

MapXtreme Java 提供了用于帮助创建 RangedTheme 或 IndividualValueTheme 的向导。向导是一个 JavaBean，可将其添加到 MapToolbar 中以便于使用。请参阅第 288 页的*使用分析图层* 。

用于专题的一般对象

可用于专题的一般对象包括 ThemeList、 Theme 和 Rendition。后续内容中对这些对象分别作出了说明。

ThemeList

ThemeList 集合可从 Layer 对象访问，并且包含 Theme 对象。 ThemeList 集合具有用于执行从集合添加、删除和重排 Theme 对象等操作的方法。

ThemeList 允许对一个图层存在多个专题影线表示。这些专题需要保持正确的顺序才可以显示在地图上。和图层相同，专题按照倒序自下而上渲染。例如，两个专题均更改了对象的填充颜色，在渲染地图时，其中的一个对象将使另一个对象变得模糊。 ThemeList 顶部的专题优先。这一排序仅限在一个图元的上下文中有效，而不是在整个图层有效。

ThemeList 对于特定情况尤为实用。例如，如果要根据人口和文化水平，以影线表示全球各个国家，那么即可采用 ThemeList。此时可以先根据人口数据，创建要以影线表示的专题，为特定区域使用不同填充颜色，然后再将其添加到 ThemeList。随后，可以根据文化程度数据，创建要以影线表示的专题，为特定区域使用有待填充的图案，然后再将其添加到 ThemeList。即使只有一个专题，也可以使用 ThemeList 对象。

Theme

Theme 是一个接口。此接口的方法可用于检索专题所基于的列和专题的说明性名称。如果用于特定专题的列或列名并不存在，则将返回空值。

Rendition

Rendition 对象的使用贯穿整个 MapXtreme。该对象为图元提供了所有样式特征。在创建专题地图时，将使用 Rendition 来指定对象的外观。

OverrideTheme

OverrideTheme 可用于更改整个图层的渲染样式。例如，如果要使用红色填充图案和绿色线条来显示 World 表，则需要使用到 OverrideTheme。

要制作用于图层的 OverrideTheme，只需在 Rendition 对象的构造器中传递 Rendition 对象即可。

```
// Assume myLayer is a Layer object.
// Assume myRend is a Rendition object.

OverrideTheme myOTheme = new OverrideTheme(myRend,"My Theme");
myLayer.getThemeList.add(myOTheme);
```

RangedTheme

RangedTheme 是一种更加复杂的专题地图。在创建范围专题地图时，所有图元均将划分到不同的范围中，每个范围均指定有与其范围相应的样式。例如，假定有一个供电视查看地区的气象台表，现在需要根据其报告的降雪量以影线表示位置。

使用范围地图图元， MapXtreme 将降雪量分为不同范围的组。如在过去的一个月中，所有降雪量为 0 至 5 英寸的气象台均划分到一个范围之内。降雪量为 5 至 10 英寸的气象台划分到另一个单独范围。降雪量为 10 至 15 英寸的气象台划分到第三个范围，而降雪量超过 15 英寸的气象台则划分到第四个范围。每个范围均称为一个 Bin。每个 Bin 均有上限和取舍值。

图层中的所有记录均将指定到相应范围，并在随后根据该范围来指定样式。例如，降雪量超过 15 英寸的气象台将表示为红色。其他范围将以比红色浅的影线表示，其中最后一个范围以灰色表示 （默认色）。在显示地图时，不同位置降雪量多少通过颜色的对比将会一目了然。

注： 如果客户端 VM 在创建较大的表上的范围专题绘图时用尽内存，则可能需要将堆的大小上限设置为 64MB。用于这一操作的命令行语法为：`Java -Xmx64m <classname>`.

MapXtreme 还提供了用于帮助创建 RangedTheme 的若干实用程序对象。

ColumnStatistics

在使用 Layer 对象的 **fetchColumnStatistics** 方法时，将返回 ColumnStatistics 对象。ColumnStatistics 对象包含有关列数据的最大值、最小值、均值和标准偏差。在使用 **fetchColumnStatistics** 方法时，将传递以影线显示地图所基于的列。此时无需直接使用 ColumnStatistics 对象的方法来创建 RangedTheme。在检索到该对象之后，该对象将用于 Bucketer 对象之中，以创建分点值的向量。

Bucketer

Bucketer 类负责计算 RangedTheme 中的 Bin 的分点值。继续以上的降雪量示例，我们有 4 个表示气象台的范围：0-5 英寸、 5-10 英寸、 10-15 英寸和 15 英寸以上的年降雪量。每个范围均为一个 Bin。

Bucketer 使用 **computeDistribution** 方法计算这些 Bin 的分点值。这些方法均返回分点值的向量。每个向量值均为一个属性对象。所有 **computeDistribution** 方法均传递范围的数量和一个 ColumnStatistics 对象。此外还可以传递分布类型和 RoundOff 对象。

分布类型

DISTRIBUTION_TYPE_EQUAL_COUNT 令每个范围的记录数目大致相同。如果要 Bucketer 将 100 条记录等分为 4 个范围，则它将计算范围，以便每个范围有大约 25 个记录，具体取决于设置的舍入因素。

在使用等计数 （或任意其他范围方法）时，需要注意可能影响到专题地图的任意特殊数值（这些值在统计学中称为*异常值*）。例如，如果使用如下数据库根据等计数方法进行影线表示：

John	5000	Andrea	7000
Penny	6000	Kyle	5500
Miguel	4500	Angela	7500
Ben	100	Mark	7000

Ben 和 Miguel 划分到同一范围 （因为两者具有两个最小值）。这样做可能不会产生所需的结果，因为 Ben 的值与其他值相去甚远。

DISTRIBUTION_TYPE_EQUAL_RANGES 将记录划分为大小相同的范围。例如，表中某一字段的数值为从 1 至 100。此时需要使用 4 个等分范围来创建专题地图。Bucketer 生成的范围为 1-25、 26-50、 51-75 和 76-100。

切记 Bucketer 可以在没有数据记录的情况下创建范围，具体取决于数据的分布。例如，如果使用以下数据库根据等范围以影线表示：

John	100	Andrea	90
Penny	6	Kyle	1
Miguel	4	Angela	92
Linda	95	Elroy	89
Ben	10	Mark	10

Bucketer 将创建 4 个范围为 1-25、 26-50、 51-75 和 76-100。注意，实际上只有两个范围（1-25 和 76-100）包含记录。

DISTRIBUTION_TYPE_STANDARD_DEVIATION 在均值的中间范围处中断，而中间范围之上 / 下的范围将是一个位于该均值之上 / 下的标准偏差。

DISTRIBUTION_TYPE_NATURAL_BREAK 根据使用每个范围平均值的算法来创建范围，以便将数据更加平均地分配于各个范围。值的分布令每个范围的平均值与该范围内的每个范围值尽可能地接近。这确保了平均值可以很好地表示其所在的范围，并且该范围内的数据值相互之间比较接近。

RoundOff

RoundOff 对象用于创建范围的规则分点值。例如，假定正在使用从 101 至 397 的值以影线表示地图，则如果范围为 100 至 400，那么会更加符合规则。 RoundOff 可以舍入至范围内的较低值，也可以舍入至范围外的较高值。

LinearRenditionSpreader

使用渐变值样式来表示数据是创建实用的专题地图的重要组成部分之一。 RangedTheme 介绍中的示例讨论以影线表示降雪量的方式。值的一端使用红色表示，接下来的一个范围使用较浅的红色表示，依此类推。 LinearRenditionSpreader 的 **spread** 方法将返回一个样式向量，从给定样式展开至另一个样式，直至达到给定元素数量。元素数量应该与传递到 Bucketer 对象的范围数量相匹配。例如，如果传递一个红色填充的样式，一个白色填充的样式，数量为 5，那么 LinearRenditionSpreader 将使用红色填充作为起始，白色填充作为结束，创建一个具有 5 个样式的向量，中间为平均展开过渡的填充类型。

创建 RangedTheme

以下示例展示了用于创建 RangedTheme 的代码，其中美国各州的表根据人口数量，使用从黄色至红色的 5 个范围以影线表示。

```
FeatureLayer lyr=null;
Rendition yellow=new RenditionImpl();
Rendition red=new RenditionImpl();
lyr = (FeatureLayer)m_map.getLayers().getLayer("States.tab");
String colName = "Pop_1990";
ColumnStatistics colStats = lyr.fetchColumnStatistics(colName);

// Set number of breaks for data
int numBreaks=5;

// Compute the distribution of data with 5 breaks and
// Equal Ranges
List rBreaks = (List) Bucketer.computeDistribution
    (numBreaks,colStats,Bucketer.DISTRIBUTION_TYPE_EQUAL_RANGES);

// Set up a red and a yellow rendition and then
// spread the colors
yellow.setValue(Rendition.FILL, Color.yellow);
yellow.setValue(Rendition.STROKE_WIDTH, 2);
red.setValue(Rendition.FILL, Color.red);
red.setValue(Rendition.STROKE_WIDTH, 4);
List rends = (List) LinearRenditionSpreader.spread
    (numBreaks, yellow, red);

// Create Theme object
RangedTheme rTheme = new RangedTheme
    (colName, rBreaks, rends, "States by Pop_1990");

// Get ThemeList class object
ThemeList tList=lyr.getThemeList();

// Add theme to Layers themeList
tList.add(rTheme);
```

范围专题还可以通过 AddTheme 向导构建。此外还可以创建关联的专题图例。

指定定制范围

通过指定定制的范围，还可创建 RangedTheme。为此需要执行以下操作：

1. 为每个范围颜色创建样式，然后添加至向量。
2. 创建表示范围上限 （分点值）的属性列表。

   注：　　该属性必须采用数字 / 字母顺序。

3. 实例化 RangedTheme 对象，指定为参数：专题列、分点值列表、样式列表和说明性名称。
4. 将该专题添加到 ThemeList。
5. 渲染地图。

以下代码示例创建三个范围：

```
// Set up the ranges
List rBreaks = new ArrayList();
rBreaks.add(new Attribute(1));
rBreaks.add(new Attribute(5));
rBreaks.add(new Attribute(7));

// Set up the renditions
List rends= new ArrayList();
rends.add(redRendition);
rends.add(grayRendition);
rends.add(greenRendition);

// Create a ranged theme object
RangedTheme rTheme = new RangedTheme(
    themeCol, rBreaks, rends, "States by Pop_1994");

// Assign theme to layers as element 1
myMap.getLayers().get(1).getThemeList().add(rTheme);

// Draw the map
// Create an ImageRequestComposer
ImageRequestComposer imageRC =  ImageRequestComposer.create(
    myMap, 256, Color.blue, "image/gif");

// Create a MapXtremeImageRenderer
MapXtremeImageRenderer renderer = new MapXtremeImageRenderer(
    "http://stockholm:8080//mapxtreme47/servelet/mapxtreme");

// Render the map
Renderer.render(imageRC);

// Render the map to the file
Renderer.toFile("comp.gif");
```

SelectionTheme

SelectionTheme 将样式应用于选择对象中引用的所有图元。此类专题通常用于存储在使用 add(FeatureSet fs) 方法的图层上使用搜索方法返回的图元。 SelectionTheme 将取代过时的 IDSelectionTheme。

例如，给定 FeatureLayer 对象和 X 以及 Y 坐标，以下代码将展示在指定位置选择图层图元，以及创建 SelectionTheme 以红色显示所选图元的过程。

```
// Assume layer as a FeatureLayer object
ArrayList columns = new ArrayList();
DoublePoint dp = new DoublePoint(x, y);
FeatureSet fs = null;

// Select the features at the specified location
fs = layer.searchAtPoint(columns, dp, null);

// Create a SelectionTheme
SelectionTheme selTheme = new SelectionTheme("PointSelection");

// Create a Selection object, and add the selected features
Selection sel = new Selection();
sel.add(fs);

// Assign the Selection object to the SelectionTheme
selTheme.setSelection(sel);

// Assign the display style of the SelectionTheme
Rendition rend = new RenditionImpl();
rend.setValue(Rendition.FILL, Color.red);
selTheme.setRendition(rend);

// Add the SelectionTheme to the layer's list of themes
layer.getThemeList().add(selTheme);
```

IndividualValueTheme

此类专题允许基于指定列的特定属性值，对图元进行专题影线表示。例如，基于 Territories 列中的值，使用单值专题来修改 Coverage 表中的区域图元的样式。带有 SouthWest 的 Territories 可以表示为红色，而带有 SouthEast 值则可以表示为蓝色。

要创建 IndividualValueTheme，请遵循以下代码示例：

```
// get a reference to the layer we will be applying theme to
FeatureLayer lyr = (FeatureLayer)lyrs.addLayer(
    dpr, ttdh, "Territories");

// create a new theme object
IndividualValueTheme iValThm = new
IndividualValueTheme("CoverageTerritory", "Sales Coverage Breakdown");

// create a rendition
Rendition rend= new RenditionImpl();

// assign color to rendition add attribute to theme with previously set
//rendition
rend.setValue(Rendition.FILL, Color.red);
iValThm.add(new Attribute("SouthWest"), rend);

// assign color to rendition add attribute to theme with previously set
//rendition
rend.setValue(Rendition.FILL, Color.blue);
iValThm.add(new Attribute("SouthEast"), rend);

// assign color to rendition add attribute to theme with previously set
//rendition
rend.setValue(Rendition.FILL, Color.green);
iValThm.add(new Attribute("Central"), rend);

// Add the theme to layers theme list
lyr.getThemeList().add(iValThm);

// Store column name and type in hashtable
Hashtable ht = new Hashtable();
ht.put("geomtype", IndividualValueThemeLegend.REGION_GEOMETRY);
ht.put("lableorder", IndividualValueThemeLegend.ORDER_ASCENDING);

// Create new legend passing theme and hashtable. The set Theme Title.
IndividualValueThemeLegend iValThmLeg;
iValThmLeg = new IndividualValueThemeLegend(iValThm, ht);
iValThmLeg.setTitle("Coverge Territory legend");

// Generate gif image from legend
iValThmLeg.toFile("c:\\temp\\terrLeg.gif", "image/gif");
```

IndividualValueTheme 也可通过 AddTheme 向导构建。此外还可以创建关联的专题图例。

专题图例

基于该数据可为范围或单值专题创建图例。对于图例的外观，可采用众多的控制选项。例如更改标题、字体，插入新的信息以及更改说明文本和颜色等。

图例可以在客户机或服务器端使用，并导出为图像（如 GIF 或 JPEG）。专题图例是可以和 MapXtreme 調 JavaBeans 一起使用的 Swing 组件。

为启用专题图例，产品增强了 Theme 接口的功能，以支持专题的 Legend 对象。每个专题都具有关联的 ThemeLegend 对象，相应对象起初均为空。只有 RangedThemes 和 IndividualValueThemes 包含的图例具有实际的信息。（其他专题类型 OverrideThemes 和 SelectionThemes 均返回空图例。）要设置与专题关联的图例，则必须调用 Theme 对象的 **setLegend(ThemeLegend)** 方法。 ThemeLegend 可通过创建一个简明方法或通过专题的 **createDefaultLegend()** 方法来获取。

以下示例展示了用于创建 RangedThemeLegend 的代码：

```
// Create Theme object
// Assume rends as a List of Rendition
// Assume colName as a attributeName(String)
// Assume rBreaks as a List of breakPoints
RangedTheme rTheme = new RangedTheme(
    colName, rBreaks, rends, "States by Pop_1990");

// Create a default legend
RangedThemeLegend rThmLeg = rTheme.createDefaultLegend(null);

// OR, Create a theme legend instance using theme and setting hashtable
// Add theme settings to hashtable
Hashtable ht = new Hashtable();
ht.put("geomtype", RangedThemeLegend.REGION_GEOMETRY);
ht.put("lableorder", RangedThemeLegend.ORDER_ASCENDING);
RangedThemeLegend rThmLeg = new RangedThemeLegend(rTheme, ht);

// Set legend title
rThmLeg.setTitle("Ranged Theme legend");

// send legend to image file
rThmLeg.toFile("c:\\temp\\rangeLeg.gif", "image/gif");
```

LegendContainerBean 管理图例的布局和显示方式。可以放入到伴随的 VisualMapJ 对象中，然后显示添加到 VisualMapJ 地图的任意图例。 VisualMapJ 通知 LegendContainerBean 何时专题发生更改，以便 LegendContainerBean 对其显示作出相应的更新。

LegendContainerBean 在作为样例 applet 的 SimpleMap 中展示。有关详细信息，请参阅第 109 页第 7 章的 *MapXtreme JavaBeans* 。

制图图例

在专题图例为专题中的样式提供键的同时，制图图例还为任意 FeatureSet 的样式集提供了键。此时将使用 com.mapinfo.legend.CartographicLegend 类来创建制图图例。既可以一次性植入 CartographicLegend 的整个 FeatureSet (addFeatureSet(FeatureSet fs, String descriptionCol) 方法），也可以每次 （通过 addRow(Rendition rend, String description) 方法）植入一个 Feature (Rendition)。

以下代码展示如何创建用于地图中的地标图层的制图图例。

```
// Assume mapJ is already initialized
FeatureLayer landmarks =
    (FeatureLayer)mapJ.getLayers().getLayer("landmarks");

// create the legend with title "State Landmarks"
CartographicLegend legend = new CartographicLegend("State Landmarks");

/* Perform a searchAll() on the layer to get its entire FeatureSet -
need to make sure we fetch the column that
*/ we want to label with in the legend ("Name")
ArrayList columns = new ArrayList();
columns.add("Name");
FeatureSet fs = landmarks.searchAll(columns, null);

/* add the FeatureSet to the legend, and specify that we
   want to label each feature's rendition with the value
*/ in the "Name" column
legend.addFeatureSet(fs, "Name");

// always dispose of the FeatureSet once you're done with it!
fs.dispose();
```

使用分析图层

MapXtreme Java 提供了包括饼图、并行条形图和堆叠条形图的功能。并行条形图垂直显示，而堆叠条形饼图则水平显示。这些图均通过 AnalysisLayer 类表示。有关详细信息，请参阅 HTML Javadoc。

条形图地图

与一个变量的值范围或渐变符号专题地图不同的是，在条形图地图中可一次查看一行中的多个变量。用户可以为每个地图对象 （图元）构建条形图，以便可以通过比较条块高度，分析特定图形中的专题变量。此外还可以在地图中查看所有图形中的同一变量。

例如，假定有一个表示美国各州的表，其中包含男女人口数量。使用条形图，即可创建专题地图，用两个条块显示每个州的人口：一个条块代表女性人口，另一个则表示男性人口。此外还可以比较每个州的人口区别，或者可以查看几个州并比较某个州和其他州之间的异同。

饼图地图

使用饼图的地图绘制分析可以一次查看同一行上的多个变量。正如比较条形图中条块的高度一样，在饼图中还可以比较单个切片的份额，或是检查特定切片在所有饼图上的份额。使用饼图，也可比较一个整体的不同组成部分。

饼图和条形图对于分析人口统计学数据尤为实用。例如，假定具有美国的人口统计信息的数据集。该数据集显示了若干人口统计群组的人口。使用饼图，可以显示各人口统计群组的人口，然后查看构成各个饼图的组成部分所占的比例大小。这样即可了解到各州或整个美国的人口统计群组的分布。此外还可以查看一个人口统计群组，观察该群组的人口和其他州不同的原因所在。

以下代码示例是供创建饼图地图参考的指南。

```
// add a pie chart analysis layer based on the world layer

// get a reference to the previously added world layer
FeatureLayer world = (FeatureLayer)mapj.getLayers().get("world");

// build a List of columns in world to analyze
List cols = new ArrayList(3);
    cols.add("Pop_0_14");
    cols.add("Pop_15_64");
    cols.add("Pop_65Plus");

    // create our AnalysisTableDescHelper - use world's
    // TableDescHelper
    AnalysisTableDescHelper atdh = new AnalysisTableDescHelper(
        world.getTableDescHelper(), cols,
        AnalysisLayerType.PIE_CHART);

    // insert the pie chart layer as the top-most layer in the
    //map - use world's DataProviderRef
    AnalysisLayer analysis =
```

```
    (AnalysisLayer)mapj.getLayers().insertLayer(
        world.getDataProviderRef(), atdh, 0, "World Population
        Analysis");

// modify some of the PieChartProperties
PieChartProperties pcp =
    PieChartProperties)analysis.getAnalysisProperties();

// define a list of custom fill styles for each wedge
List wedgeStyles = new ArrayList(3);
Rendition red = new RenditionImpl();
red.setValue(Rendition.FILL, Color.red);
wedgeStyles.add(red);
Rendition yellow = new RenditionImpl();
yellow.setValue(Rendition.FILL, Color.yellow);
wedgeStyles.add(yellow);
Rendition orange = new RenditionImpl();
orange.setValue(Rendition.FILL, Color.orange);
wedgeStyles.add(orange);
pcp.setSectionFillStyles(wedgeStyles);

// use a 2-pixel wide line style for the pie chart
// border and wedge dividers
Rendition lineStyle = new RenditionImpl();
lineStyle.setValue(Rendition.STROKE_WIDTH, 2);
pcp.setLineStyle(lineStyle);

// start drawing the first wedge at 0°, and work
// counter-clockwise from there
pcp.setStartAngle(0);
pcp.setDrawnClockwise(false);
```

第 3 部分：高级主题

本开发人员指南的第 3 部分为开发人员提供了 MapXtreme Java 中高级特性的有关信息。

主题：

XML 协议

16

本章介绍 MapInfo Enterprise XML 协议。

本章内容：

MapInfo Enterprise XML 协议

MapXtreme Java 内建支持 MapInfo Enterprise XML 协议，可协同各种企业应用程序工作。此协议基于扩展标记语言 (XML)，定义众多 MapInfo 企业产品如何处理对于地图信息和数据的请求和响应，例如 MapXtreme Java、 MapMarker J Server 和 Routing J Server 等产品。

目前， MapInfo Enterprise XML 协议提供两个支持的组件：**MapImageRequests** 以及本版本的新组件 **MapVectorRequests**。

MapImageRequest 是一个提交至 MapXtremeServlet 用于请求多图层地图渲染的 XML 文档。此外， MapImageRequest 也用于请求生成动画图像的基础地图以及点覆盖列表。有关详细信息，请参阅第 296 页的 *MapImageRequest* 。

借助于 MapVectorRequest 协议，使用 MapXtreme Java 类创建的所有客户机和非 MTXJ 客户机即可作为图层搜索的一部分创建 MapVectorRequest，并以地理标记语言 (GML) 格式接收图层的原始数据。有关 MapVectorRequests 的详细信息，请参阅第 299 页的 *MapVectorRequest* 。

GML 是由开放式 GIS 协会 (OGC) 所开发的 XML 编码，用于说明地理图元的属性和几何特征。正如 XML 将 Web 内容与显示隔离开来一样， GML 在说明图元几何特征的同时，将其与图形解释和图元的可视化元素隔离开来。

文档类型定义 (DTD)

MapXtreme Java 中还包括了文档类型定义 (DTD)， DTD 指定一组 XML 请求和响应文档的结构规则。DTD 定义了每个有效元素的语法。在安装之后，DTD 位于 MapXtreme-4.7.0/lib 目录的 mxjdtds.jar 文件之中。务必确保该 jar 文件位于 classpath 中。

定义 DTD 元素的 HTML 文档随 MapXtreme Java 提供，其位置为 /docs/xmlprot/index.htm。

注： 同时支持 4.0 版和 4.5 版的 DTD。

支持的 **DTD**	说明
MI_XML_Protocol_MapImageRequest_4_7.dtd	定义客户机和服务器之间用于地图图像的请求和响应。
MI_XML_Protocol_MapVectorRequest_4_7.dtd	以地理标记语言 (GML) 格式定义客户机和服务器之间用于向量数据的请求和响应。

在 MapImageRequest 或 MapVectorRequest 中支持使用以下 DTD 的内容。

独立于产品的 DTD

- MI_XML_Protocol_CommonElements_1_0.dtd：
  定义时间、距离、角度和速度等度量单位。此外还定义国际化的元素。其对于线路规划、地理编码和地图绘制的支持均独立于产品。

MapXtreme Java DTD

- MI_XML_Protocol_MapCommonElements_4_7.dtd：
  定义所有 MapXtreme 请求和响应公共的地图元素。说明支持 MapVectorResponses 的新 GMLFeature 和 GMLFeatureSetElements。
- MI_XML_Protocol_MapFeatureStyle_4_7.dtd：
  定义说明用于图元或专题样式的元素。
- MI_XML_Protocol_MapStyle_4_7.dtd：
  定义样式元素，例如填充、单笔填充、字体和标记等。
- MI_XML_Protocol_MapTextLabels_4_7.dtd：
  定义说明标准的元素，其中包括基础属性、样式、专题、位置和约束条件 （如是否允许标准重复或覆盖）。

OGC 特定的 DTD

开放式 GIS 协会 (OGC) 在规范中对这些 DTD 作出了概要介绍。 DTD 中采用了完全相同的规范。

- MI_XML_Protocol_OGC_CRS_1_1.dtd：
  OGC 坐标参考系定义。
- MI_XML_Protocol_OGC_GML_1_0.dtd：
  定义用于点、多边形、折线和集合的简单 GML （地理标记语言）几何元素，符合 OGC 定义的规范。
- MI_XML_Protocol_OGC_Identification_1_0.dtd：
  定义 OGC 标识元素。
- MI_XML_Protocol_OGC_Units_1_1.dtd：
  定义时间、距离、角度、像素间距度量的 OGC 单位。

MapImageRequest

MapImageRequest 是一个用于请求 MapXtreme Java 渲染多图层地图的 XML 文档。MapImageRequest 包含服务器访问每个图层中的数据、要应用于每个图层几何对象的专题影线表示定义、要为每个图层生成的标注定义，以及所要返回的图像属性的定义所需的全部信息。

DTD 文件 MI_XML_Protocol_MapImageRequest_4_7.dtd 定义了 MapXtreme 4.5 MapImageRequest、MapImageResponse 和 MapImageFaultResponse 的语法。此文档使用了外部 !ENTITY 结构以包含另外 9 个 DTD 文件，这 9 个文件元素定义在其他 XML 协议请求之间共享。上述所有 DTD 文件均可见于随 MapXtreme Java 4.7 版本发布的 midtds.jar 文件之中。

MapImageRequest 可通过采用任何语言编写的、在任意平台上运行的任意客户机应用程序发送到 MapXtreme Java 服务器，只要其遵循该协议即可。对于客户机而言，不必使用 MapXtreme Java 客户机类实施客户机 （MapJ 和 关联类）。

创建和发送 MapImageRequests 对于编写其自己的客户机，并直接从 / 至 MapXtreme Java 服务器处理请求 / 响应的应用程序尤为重要。

创建 MapImageRequest

不使用任意 MapXtreme Java 客户机类 （MapJ 和关联的类）的客户机应用程序必须通过其自己的途径，并以符合本文所述控制 XML 文档内容的 DTD 语法和其他策略来创建 MapImageRequest。

要使用 MapXtreme Java 客户机类的客户机应用程序可以创建 MapJ 对象，并使用 ImageRequestComposer 类从 MapJ 对象创建 MapImageRequest。由 ImageRequestComposer 生成的文档在发送到服务器之前无需更改。

将 MapImageRequest 发送到服务器

MapImageRequest 将使用 HTTP 1.1 POST 请求发送到服务器。

```
HTTP 1.1
    Request-Line: POST
    Content-Language: en_us
    Content-Type: text/xml; charset=iso-8859-4
    Accept-Language: en_us
    Accept-Charset: iso-8859-4

    MI_XMLProtocolRequest:MapImageRequest
    MI_XMLProtocolVersion: MI_XML_Protocol_MapImageRequest_4_7
```

```
MI_XMLProtocolTransactionId: client_defined_id

<RequestEnvelope>
    <MapImageRequest>

    //request content

    </MapImageRequest>
</RequestEnvelope>
```

该协议包括 3 个定制的 HTTP 标头。其中两个为必需，第三个为可选。

- **MI_XMLProtocolRequest** – 必需。
  确定协议请求类型。此例中为字符串 MapImageRequest。
- **MI_XMLProtocolVersion** – 必需。
  确定请求的协议版本。尤为重要的是，这是定义请求语法的 DTD 版本。此版本字符串在 DTD 文件本身的标头注释中确定。例如，MI_XML_Protocol_MapImageRequest_4_7.dtd 文件包含确定此版本的标头注释，如下所示。

```
<!-- ********************************************** -->
<!-- ********************************************** -->
<!--The MI_XMLProtocolVersion of this DTD is:
    MI_XML_Protocol_MapImageRequest_4_7 -->
<!-- ********************************************** -->
<!-- ********************************************** -->
```

- **MI_XMLProtocolTransactionId** – 可选。
  由客户机应用程序定义。用于关联响应与请求。

从 MapImageRequest 接收响应

源自 MapImageRequest 的响应如下：一个图像、一个包含在 XML MapImageResponse 中的 base64 编码图像或 XML MapImageFaultResponse。

返回图像的成功示例：

```
HTTP 1.1
    Content-Type: image/gif
    MI_XMLProtocolTransactionId: client_defined_id
```



在返回图像的成功示例中， HTTP 响应不是 XML 文档，只是图像。

标头 MI_XMLProtocolTransactionId 为可选，并且只有在原始请求中出现时包括。

成功返回 base64 编码图像的成功示例：

```
HTTP 1.1
    Content-Language: en_us
    Content-Type: text/xml; charset=iso-8859-4
    MI_XMLProtocolResponse: MapImageResponse
    MI_XMLProtocolVersion: MI_XML_Protocol_MapImageRequest_4_7
    MI_XMLProtocolTransactionId: client_defined_id


    <ResponseEnvelope>
        <MapImageResponse>
            <EncodedImage Encoding="base64"
            MimeType="image/gif">
            . . .
            </EncodedImage>
        </MapImageResponse>
    </ResponseEnvelope>
```

在返回编码图像的成功示例中， HTTP 响应是一个包含表示为编码文本的图像的 XML 文档。标头 MI_XMLProtocolResponse 和 MI_XMLProtocolVersion 为必需，标头 MI_XMLProtocolTransactionId 为可选，并且只有在原始请求中出现时包括。

所有失败的示例：

```
HTTP 1.1
    Content-Language: en_us
    Content-Type: text/xml; charset=iso-8859-4
    MI_XMLProtocolResponse: MapImageFaultResponse
    MI_XMLProtocolVersion: MI_XML_Protocol_MapImageRequest_4_7
    MI_XMLProtocolTransactionId:client_defined_id
```

```
<ResponseEnvelope>
    <MapImageFaultResponse>

   //response content

    </MapImageFaultResponse>
</ResponseEnvelope>
```

所有失败的示例均返回 XML MapImageFaultResponse。标头 MI_XMLProtocolResponse 和 MI_XMLProtocolVersion 为必需，标头 MI_XMLProtocolTransactionId 为可选，并且只有在原始请求中出现时包括。

MapVectorRequest

MapVectorRequest 是一个 XML 文档，请求 MapXtreme Java 服务器对数据源执行搜索，返回作为图元集的结果中包括属性、样式和几何数据。术语 “向量”特指将几何数据表示为点集的事实。 MapVectorRequest 确定数据源、搜索条件和要返回的特定数据项。

DTD 文件 MI_XML_Protocol_MapVectorRequest_4_7.dtd 定义 MapXtreme 4.7 MapVectorRequest、 MapVectorResponse 和 MapVectorFaultResponse 的语法。此文档使用了外部 !ENTITY 结构以包含另外 9 个 DTD 文件，这 9 个文件元素定义在其他 XML 协议请求之间共享。所有的 DTD 文件均可见于随 MapXtreme Java 4.7 版本发布的 midtds.jar 文件之中。

MapVectorRequest 可通过采用任何语言编写的、在任意平台上运行的任意客户机应用程序发送到 MapXtreme Java 服务器，只要其遵循该协议即可。对于客户机而言，不必使用 MapXtreme Java 客户机类实施客户机 （MapJ 和 关联类）。

创建和发送 MapVectorRequests 对于编写其自己的客户机，并需要直接从 / 至 MapXtreme Java 服务器处理请求 / 响应的应用程序尤为重要。

创建 MapVectorRequest

不使用任意 MapXtreme Java 客户机类 （MapJ 和关联的类）的客户机应用程序必须通过其自己的途径，并以符合本文所述控制 XML 文档内容的 DTD 语法和其他策略来创建 MapVectorRequest。

要使用 MapXtreme Java 客户机类的客户机应用程序可以创建 MapJ 对象，并使用 LayerVectorRequestComposer 类从图层对象创建 MapVectorRequest。由 LayerVectorRequestComposer 生成的文档在发送到服务器之前无需更改。

将 MapVectorRequest 发送到服务器

MapVectorRequest 将使用 HTTP 1.1 POST 请求发送到服务器。

```
HTTP 1.1
    Request-Line: POST
    Content-Language: en_us
    Content-Type: text/xml; charset=iso-8859-4
    Accept-Language: en_us
    Accept-Charset: iso-8859-4

    MI_XMLProtocolRequest:MapVectorRequest
    MI_XMLProtocolVersion: Mi_XML_Protocol_MapVectorRequest_4_7
    MI_XMLProtocolTrasactionId: client_defined_id


    <RequestEnvelope>
        <MapVectorRequest>

        //request content

        </MapVectorRequest>
    </RequestEnvelope>
```

该协议包括 3 个定制的 HTTP 标头。其中两个为必需，第三个为可选。

- MI_XMLProtocolRequest – 必需。确定协议请求类型。此例中为字符串 MapVectorRequest。
- MI_XMLProtocolVersion – 必需。确定请求的协议版本。尤为重要的是，这是定义请求语法的 DTD 版本。此版本字符串在 DTD 文件本身的标头注释中确定。例如，MI_XML_Protocol_MapVectorRequest_4_7.dtd 文件包含确定此版本的标头注释，如下所示。

```
<!-- ********************************************** -->
<!-- ********************************************** -->
<!--The MI_XMLProtocolVersion of this DTD is:
    MI_XML_Protocol_MapVectorRequest_4_7              -->
<!-- ********************************************** -->
<!-- ********************************************** -->
```

- MI_XMLProtocolTransactionId – 可选。由客户机应用程序定义。用于关联响应与请求。

从 MapVectorRequest 接收响应。

源自 MapVectorRequest 的响应如下：XML MapVectorResponse 或 XML MapVectorFaultResponse。

返回图像的成功示例：

```
HTTP 1.1
    Request-Line: POST
    Content-Language: en_us
    Content-Type: text/xml; charset=iso-8859-4
    Accept-Language: en_us
    Accept-Charset: iso-8859-4

    MI_XMLProtocolRequest: MapVectorResponse
    MI_XMLProtocolVersion: MI_XML_Protocol_MapVectorResponse_4_7
    MI_XMLProtocolTrasactionId: client_defined_id

    <ResponseEnvelope>
        <MapVectorResponse>
            <MIFeatureSet>
             . . .
            </MIFeatureSet>
        </MapVectorResponse>
    </ResponseEnvelope>
```

在成功的示例中， HTTP 响应是一个包含 MIFeatureSet 的 XML 文档。标头 MI_XMLProtocolResponse 和 MI_XMLProtocolVersion 为必需，标头 MI_XMLProtocolTransactionId 为可选，并且只有在原始请求中出现时包括。应该注意 XML 响应文档可能会非常大，具体大小取决于在数据源中出现的几何对象的数量和类型。

失败的示例返回 XML MapVectorFaultResponse。标头 MI_XMLProtocolResponse 和 MI_XMLProtocolVersion 均为必需。header MI_XMLProtocolTransactionId 为可选，并且只有在原始请求中出现时包括。

```
HTTP 1.1
    Content-Language: en_us
    Content-Type: text/xml; charset=iso-8859-4
    MI_XMLProtocolResponse: MapVectorFaultResponse
    MI_XMLProtocolVersion: MI_XML_Protocol_ MapVectorResponse_4_7
    MI_XMLProtocolTransactionId: client_defined_id

    <ResponseEnvelope>
        <MapVectorFaultResponse>

        //response content

        </MapVectorFaultResponse>
    </ResponseEnvelope>
```

17 自定义 AddLayer 向导

本章介绍如何自定义 AddLayer 向导。

本章内容：

概览

增加图层向导是一个帮助用户添加图层的向导工具。在 “图层控制”对话框中单击 “添加”按钮，即可显示该向导。要获取 “增加图层向导”行为的示例，可运行 MapXtreme Java 管理器，然后单击 “图层控制”按钮。从 “图层控制”对话框，单击 “添加”按钮，以显示 “增加图层向导”。

“增加图层向导”将根据存储在 **addlayerwizard.properties** 文件中的数据源的值自行初始化。该文件为文本文件，由 MapXtreme Java 安装程序安装到 MapXtreme Java 安装目录下的 lib/client 目录中。该文件中的大部分行都包括一个键和一个与之关联的值，其间由等号 (=) 间隔。此文件中的值可以修改，以更改 AddLayer 向导的配置，其中包括：

- 应该出现在数据源初始列表中的数据源列表。
- 在从初始列表选择 “使用命名数据库连接”数据源 （此数据源默认为不出现）之后将出现的 “命名连接”列表。
- 提供用于向导操作步骤的数据源信息的默认值和常用值。
- 用于向导的第一步 （选择数据源）和最后一步 （指定如何访问数据）的默认值和 / 或常用值。
- 口令是否保存到属性文件 （默认为不保存）。

更改数据源列表

由 MapXtreme Java 安装的 addlayerwizard.properties 文件配置用于在向导的第一步提供以下数据源列表：

- MapInfo TAB 文件
- 有空间选项的 Oracle
- 有 SpatialWare 的 SQL Server
- 有 SpatialWare 的 IUS
- 任意具有 X/Y 列的数据
- GeoTIFF 栅格文件
- ESRI Shape
- 数据绑定
- 命名图层
- 使用命名数据库连接 （只有在手动添加命名连接之后可用）。

如果在惯用的文本文件查看器中打开 addlayerwizard.properties 文件，将会在该文件顶部看到以下行：

```
DataSource1=MapInfo TAB file
```

其中的 1 表示将成为列表中第一个数据源的 MapInfo TAB 文件数据源。随后是上述列表中的其他各个数据源 （按照数字排序）。

删除数据源

如果已知您 （或是 applet/ 应用程序的用户）将不会从上述列表的任意数据源添加图层，则可以将其从 addlayerwizard.properties 文件的列表中将其删除。但是，如果在列表的中部删除数据源，则必须调整列表中该数据源之后所有数据源的编号。例如，由于不需要从 SQL 服务器数据源添加图层，因此可以删除以下条目组。

```
DataSource3=SQL Server with SpatialWare
DataSource3_DPHelper_Page=DataSource3_Page2
DataSource3_DPHelper_Class=com.mapinfo.dp.jdbc.sqlserversw.
    SQLServerSpwDataProviderHelper
DataSource3_Page_Count=3
DataSource3_Page1=com.mapinfo.beans.addlayer.PageSQLSRVp1
DataSource3_Page1_Description=Specify SQL Server Data Source
    Information
DataSource3_Page2=com.mapinfo.beans.addlayer.PageSQLSRVp2
DataSource3_Page2_Description=Specify SQL Server Table or Query
    Information
DataSource3_Page3=com.mapinfo.beans.addlayer.PageSQLSRVp3
DataSource3_Page3_Description=Specify Other SQL Server Table or Query
    Information
```

随后，您必须在该属性文件中上移其后的所有数据源 （编号）。因此， DataSource4 条目组将更改为 DataSource3 条目， DataSource5 条目组将更改为 DataSource4 条目，依此类推。

我们建议不要从属性文件删除条目，而是对该组条目进行注释，即在该组条目的各行之前添加 # 字符。例如，对 SQL 服务器条目进行注释的示例如下所示。

```
#DataSource3=SQL Server with SpatialWare
#DataSource3_DPHelper_Page=DataSource3_Page2
#DataSource3_DPHelper_Class=com.mapinfo.dp.jdbc.sqlserversw.
    SQLServerSpwDataProviderHelper
#DataSource3_Page_Count=3
#DataSource3_Page1=com.mapinfo.beans.addlayer.PageSQLSRVp1
#DataSource3_Page1_Description=Specify SQL Server Data Source
    Information
#DataSource3_Page2=com.mapinfo.beans.addlayer.PageSQLSRVp2
#DataSource3_Page2_Description=Specify SQL Server Table or Query
    Information
#DataSource3_Page3=com.mapinfo.beans.addlayer.PageSQLSRVp3
```

```
#DataSource3_Page3_Description=Specify Other SQL Server Table or Query
    Information
```

如果只是对这些行进行注释，那么删除 # 字符，即可将相应的数据源添加回到列表之中。

此外，如果删除到只剩一个数据源，那么 AddLayerWizard 的第一步将不会显示数据源列表，因为对于一个数据源而言别无他选。

数据源重新排序

与每个数据源条目组关联的编号决定了其在数据源列表中的位置，这些数据源将在添加数据源向导的第一步中显示。

通常，我们将常用的数据源放在列表中靠前的位置上。例如，如果已知 applet/ 应用程序用户经常需要从 IUS 数据源向其地图添加图层，则可选择将该数据源置于列表首位。默认的 addlayerwizard.properties 文件表示 IUS 数据源将作为列表中的第四个数据源显示。要将其置于首位，可对条目组更改如下：

```
DataSource1=IUS with SpatialWare
DataSource1_DPHelper_Page=DataSource5_Page2
DataSource1_DPHelper_Class=com.mapinfo.dp.jdbc.iussw.
    IusSpwDataProviderHelper
DataSource1_Page_Count=3
DataSource1_Page1=com.mapinfo.beans.addlayer.PageIUSp1
DataSource1_Page1_Description=Specify IUS Data Source Information
DataSource1_Page2=com.mapinfo.beans.addlayer.PageIUSp2
DataSource1_Page2_Description=Specify IUS Table or Query Information
DataSource1_Page3=com.mapinfo.beans.addlayer.PageIUSp3
DataSource1_Page3_Description=Specify Other IUS Table or Query
    Information
```

自然，这样作将会令属性文件中出现两个 DataSource1 条目。要解决此问题，可增加和该数据源关联的编号，以便将其置于新的 1 号条目之后。

指定命名数据库连接

如果已经查看了由 MapXtreme Java 安装的 addlayerwizard.properties 文件的内容，则可能会注意到其中包括了第 11 个数据源，即使用命名数据库连接。此外，还可能注意到这一数据源并未出现在向导第一步中供选的数据源列表之内。其原因在于已经安装的 addlayerwizard.properties 文件没有任何命名连接条目。

这些命名连接条目是和由连接管理器面板管理的命名连接相同的连接 （请参阅*第 12 章：访问远程数据*），并且存储在 **miconnections.properties** 文件中。命名连接为特定数据库连接 （更为准确地说，是一组数据库属性）提供了轻松的引用方式。

要使 applet/ 应用程序的用户可以基于这些命名连接之一来添加图层，而无需指定任何连接信息，则需要向 addlayerwizard.properties 文件的命名连接列表添加相应的命名连接。

例如，如果此前已经定义名为 Enigma 的连接，该连接定义了到 Oracle Spatial 数据源的连接，则需要向 addlayerwizard.properties 文件添加一对条目，令 "添加条目向导" 意识到命名连接 / 资源的存在，参阅以下粗体文本：

```
# Named Connections
# Add any named Connections that appear in
# miconnections.properties.
# Each named resource entry should include the name of the
# resource as well
# as the data source which is appropriate for that named
# resource.
# Sample named resource:
#    NamedConnection1=snoopy
#    NamedConnection1_DataSource=DataSource2
# In this sample, DataSource2 represents the data source which
# is appropriate for snoopy.
NamedConnection1=Enigma
NamedConnection1_DataSource=DataSource2
```

在文件中紧接着命名连接注释部分之后的位置添加条目，以便可以从一个位置管理所有的命名连接。相应注释部分还说明了定义命名连接时所要使用的约定。唯一不明显的地方如下所示：

```
NamedConnection1_DataSource=DataSource2
```

此行表示 DataSource2 （默认为 Oracle Spatial 数据源）是相应于命名 Enigma 连接的数据源。

您可以根据需要添加任意数量的命名资源，前提是相应数据源和 miconnections.properties 文件中的命名连接保持一致。在列表中至少有一个命名资源之后，相应的 "命名资源" 数据源将显示在向导第一步的数据源列表之中。有关连接和命名资源的详细信息，请参阅*第 12 章：访问远程数据*。

addlayerwizard.properties 的位置

此处有两个 addlayerwizard.properties 副本。其一位于 **lib/client** 之中，在运行类似 MapXtreme Java 管理器应用程序的独立应用程序时使用。

addlayerwizard.properties 的另一个副本存储在 webapps/mapxtreme47/client 目录的 mjmappletsup.jar 文件之内，该副本在运行作为 applet 的 MapXtreme Java 管理器时使用。当作为 applet 运行 MapXtreme Java 管理器时，将会使用略有不同的安全设置，因此其需要自己的属性文件副本。

如果需要编辑 addlayerwizard.properties 文件，确保编辑的是计划运行的应用程序的适当文件副本。

指定默认值

可以置入各种 "增加图层向导" 控件的默认值，这些默认值显示为预选设置或出现相应设置时的指定默认值。这一点非常实用，尤其是在 applet/ 应用程序的用户不知道远程数据源的连接信息的情况下，因为此时您可以自动为其填充相应的数据。

要为控件指定默认值，可在 addlayerwizard.properties 中添加行，表示要设置的页面和控件（两者共同构成 "键"）以及默认值。例如，如果要设置在向导的初始 DataSourcePage 页面显示为预先选定的数据源，可以添加如下行：

```
DataSourcePage_default_datasource=Oracle with Spatial Option
```

注： 产品安装时已经安装了包含此行的 addlayerwizard.properties 文件，即指定 MapInfo TAB 文件为默认数据源。

用于在 addlayerwizard.properties 中指定默认值的约定如下所示：

```
<page>_default_<control name>=<default value>
```

注： 在默认值前后请勿包括双引号。

设置用于首页和末页控件的默认值

在向导的首页 （屏幕）中，可选择用于添加图层的数据源。而在末页中，则可指定访问数据的方式和设置某些杂项值。

要为任意页面中的控件指定默认值，可按照上述指定默认值的约定，向属性文件添加相应的行。

页面	控件名称	控件说明
DataSourcePage	datasource	“可用的数据源”列表中预先选定的数据源（如 MapInfo TAB 文件）
DataSourcePage	useprevious	如为 True 则表示预先选择 “使用先前的设置作为默认值”选项；如为 False 则表示预先选择 “使用默认属性值”选项。
DataAccessPage	servlet	如为 True 则表示预先选择 “由远程 MapXtremeServlet 访问数据”；如为 False 则表示预先选择 “访问本地数据”选项。
DataAccessPage	url	URL － MapXtremeServlet 的 URL

设置用于其他 DataSource 页面控件的默认值

“增加图层向导”中的每个数据源均具有一个或多个与其关联的 “页面”，用于查询从该类数据源添加特定图层所需的必要信息。如果查看 addlayerwizard.properties 的内容，将会注意到其中有多种 DataSourceX_PageY 条目。您所指定的默认值将为向导某一页面所专用。

增加图层向导可以保持在会话之间最常使用 (MRU) 的值集。增加图层向导具有对 addlayerwizard.properties 文件的写访问权限，在您每次单击 “完成”按钮之后，向导将写入各种 MRU 值行。如果选择了 “使用先前的设置作为默认值”选项，则这些值将在向导的首页中作为预选的值。这一特性有助于避免重复键入相同或类似的设置。

注： 如果在 applet 中部署，则增加图层向导不能写入此文件。

确认可以设置哪一默认值的最简单办法是在单击添加图层的 “完成”按钮之后，查看所保存的 MRU 值。复制现有的 MRU 行来创建其另一 （默认）版本，再使用 mru 替代 default。

保存口令

默认情况下，增加图层向导配置为不在属性文件中保存最常用的口令。在添加远程数据库中的图层时，口令是连接信息的必要组成部分。为了安全起见，默认情况下，这些口令将不保存在属性文件中。但是，如果希望将 MRU 口令保存在属性文件中，只需更改 addlayerwizard.properties 中的一行信息即可。即将属性文件的第一行更改如下：

```
Save_Passwords=true
```

18

创建定制数据提供方

本章介绍如何实施定制的数据提供方。

本章内容：

简介

在 MapXtreme Java 4.0 之前，只能使用由 MapXtreme 支持的数据源集合提供的数据提供方来检索数据。在 3.0 版中添加的 QueryBuilder 功能提供了定制 MapXtreme 数据提供方行为的机制，但是仍然将访问权限限定为支持的数据源。在 4.x 发布版中，创建数据提供方所需的全部接口和类均完全公开，用户借此即可采用任何方式来访问各种类型的数据。

实施 DataProvider 需要在开发和测试资源上完成大量工作，这不是一项轻松的任务。几乎所有用户都发现 MapXtreme DataProviders 已经能够完全满足需要。只有那些在 MapXtreme 中无法找到所需特定功能，并且具备实施 DataProvider 所需资源的用户，才有可能实施 DataProvider。

什么是 DataProvider?

MapXtreme 应用程序基于由一个或多个图层对象构成的 MapJ 对象。每个图层中的数据均取自特定的数据源 （如 TAB 文件、 RDBMS 或栅格文件）。图层本身并不识别植入数据的特定数据源、类型和访问机制。使用提供给其构造器的对象，图层对象即可创建 DataProvider 接口的实例，以执行将数据返回图层的必要操作。作为示例，如果数据源是 RDBMS 并且请求返回所有包含在边界矩形中的图元， DataProvider 将需要执行以下操作：

1. 创建包含正确几何搜索约束条件并返回必要数据的 SQL 查询字符串。
2. 建立到 RDBMS 的连接。
3. 执行查询。
4. 取回结果集并将数据转换为 MapXtreme 所请求的格式返回。

总之，DataProvider 接口需要实施采用不同约束条件和两个元数据查询的 8 个数据查询。从 DataProvider 返回的数据应该是图元数据 （向量几何和属性）或图像数据 （栅格图像）。用于这两种类型数据的 DataProvider 和相关接口的实施略有不同。下表确定了两种数据所需的接口。

主数据提供方接口和文件概览

接口	文件包	向量	栅格	说明
DataProviderHelper	com.mapinfo.dp	是	是	确定访问数据源所需的元数据。
TableDescHelper	com.mapinfo.dp	是	是	确定要在数据源处访问的数据。
DataProvider	com.mapinfo.dp	是	是	查询数据源，检索数据和元数据。
UpdateableDataProvider	com.mapinfo.dp	否*	否	扩展 DataProvider，以允许编辑数据源中的数据。
FeatureSet	com.mapinfo.dp	是	是	表示由 DataProvider 执行的查询的所有数据查询结果，即一个实施为图元的记录集合。
Feature	com.mapinfo.dp	是	是	FeatureSet 中的单一记录。
Geometry	com.mapinfo.dp	是	否	确定图元的常规几何信息。
PointGeometry	com.mapinfo.dp	是	否	将 Geometry 扩展为点数据特有的接口。
VectorGeometry	com.mapinfo.dp	是	否	将 Geometry 扩展为线数据或区域数据特有的接口。
PointList	com.mapinfo.dp	是	否	提供对 VectorGeometry 坐标数据的访问。
MIRaster	com.mapinfo.dp	否	是	将栅格数据绘制到 Java 图形对象。
TableInfo	com.mapinfo.dp	是	是	有关由 TableDescHelper （Layer 元数据）确定的数据的元数据或从 DataProvider 查询 （FeatureSet 元数据）返回的数据的元数据。

接口	文件包	向量	栅格	说明
ColumnStatistics	com.mapinfo.dp	否 犟	否	有关数据源中的单列数据的统计信息。只和 RangedTheme 或 RangedLabelTheme 一起使用。
LayerXMLHandler	com.mapinfo.xmlprot.mxtj.layerxmlhandlers	是	是	提供数据提供方类从 / 至 MI_XML_Protocol_MapCommonElements_4_7.dtd 中定义的 XML 元素的映射。
layerxmlhandlers.xml	未提供	是	是	系统配置文件，是提供在 LayerXMLHandler 和 DataProviderHelper 之间的映射的外部配置文件。

* 如果数据源为只读则不需要
犟 如果没有使用 RangedTheme 和 RangedLabelTheme 则不需要。

用于向量数据的 DataProvider 实施步骤

1. 实施 **TableDescHelper**
   实施接口方法，并在数据源中添加所有数据特定的其他接口方法。 DataProvider 实施不需要的接口方法可以不采用任何操作。 TableDescHelper 存储在图层之中，并传递到所有 DataProvider 接口方法以及 LayerXMLHandler.createQueryElement() 接口方法，因此其必须包含这些方法所需的数据。
2. 实施 **DataProviderHelper**
   此接口没有方法，因此必须为数据源添加所需的全部方法。 DataProviderHelper 存储在图层之内，并将传递到 theLayerXMLHandler.createConnectionElement() 接口方法，因此其必须包含此方法所需的数据。
3. 实施 **TableInfo** 和 **DataProvider.getTableInfo()**
   可连接到数据源并获取相应数据库所存储的数据类型的有关信息。
   DataProvider.getTableInfo() 和 FeatureSet.getTableInfo() 均需要实施 TableInfo。
4. 实施 **DataProvider.queryInRect()**
   可对数据源中的数据进行有条件的几何搜索。 MapXtreme 也需要使用这一方法来进行渲染。

5. 实施 **FeatureSet**、 **Feature**、 **Geometry**、 **VectorGeometry**、 **PointGeometry** 和 **PointList**

   通过这些对象，MapXtreme 可以使用 queryInRect() 实施的结果。对于 DataProvider 的请求表明了在 FeatureSet 中返回的几何数据所需的坐标系。因此，在返回到 MapXtreme 之前 （请参阅 com.mapinfo.coordsys.CoordSys 和 com.mapinfo.coordsys.CoordTransform），有必要转换从数据源取回的几何坐标。

6. 实施 **LayerXMLHandler**

   此接口实施 Java 类和 XML 元素之间的映射 （JDOM 元素对象）。 XML 必须符合随 MapXtreme （请参阅 midtds40.jar）提供的 DTD 中的文档定义。

   *** 有关 <Connection> 元素下的 <Url> 元素的要求 ***

   <Url> 元素中的字符串必须以您选择用于确认由 “:”（冒号字符）间隔的 DataProvider 的数据提供方协议标记开始。(colon character). 例如， MapXtreme Oracle 和 Tab 数据提供方使用 oraso 和 tab 标记，示例 XML Url 元素如下所示：

```
<Url>jdbc:oracle:thin:@your_server:1521:your_servicename</Url>
<Url>tab:d:\Program Files\maps</Url>
```

   如果实施的是 MyDataProvider，并且选择 mydp 作为协议标记，则 Url 将如下所示：

```
<Url>mydp:any_url_string@blah.blah.blah:dotcom?hello</Url>
```

   此外还可任选其他标记。配置文件 layerxmlhandlers.xml 中也确认了这一相同的标记 （如下所述）。 MapXtreme 在 <Url> 元素的开始部分查找此标记，并将其映射到适当的 LayerXMLHandler 以便实例化，读取由处理程序生成的 XML 文档的各个部分。 Url 的其余部分可以采用任意所需的格式。

7. 向 **layerxmlhandlers.xml** 添加条目

   此文件为 MapXtreme 提供了数据提供方协议标记以及 DataProviderHelper 和 LayerXMLHandler 实施的类名之间的关联。此文件必须驻留在 MapXtreme 应用程序的类路径中。此文件的模板随 MapXtreme 安装提供。

   以下是 MyDataProvider 的示例条目。在此需要使用完全合格的类名。<DataProviderProtocol> 元素是必须追加到连接 URL 的上述标记。

```
<LayerXMLHandlerMapping>
    <DataProviderHelperClass>
         com.mycompany.myapp.MyDataProviderHelper
    </DataProviderHelperClass>
<LayerXMLHandlerClass>
        com.mycompany.myapp.MyLayerXMLHandler
</LayerXMLHandlerClass>
    <DataProviderProtocol>
        mydp
    </DataProviderProtocol>
</LayerXMLHandlerMapping>
```

8. 渲染使用 **DataProvider** 的图层
   渲染完成之后， DataProvider 实施的主要障碍就不复存在了。
9. 实施其余的 **DataProvider** 接口方法
   实施 ColumnStatistics。
10. 实施 **UpdateableDataProvider** （可选）
    如果要允许编辑数据源，即与只读相反，可以更改数据提供方来实施 UpdateableDataProvider 接口并在该接口上实施附加方法。

用于栅格数据的 DataProvider 实施步骤

1. 与此前用于向量数据的步骤 1 相同。
2. 与用于向量数据的步骤 2 相同。
3. 与用于向量数据的步骤 3 相同。由于栅格数据的元数据很少，因此 DataProvider 实施无需采用的 TableInfo 接口方法可不采用任何操作。
4. 扩展 com.mapinfo.dp.util.AbstractRasterDataProvider。
5. 与用于向量数据的步骤 6 相同。
6. 与用于向量数据的步骤 7 相同。
7. （可选）覆盖 com.mapinfo.dp.util.AbstractRasterDataProvider.getValue(TableDescHelper, DoublePoint). 在实施读取栅格 （如 MIG、 GRD）的数据提供方时，只需覆盖此方法。

注： 扩展 AbstractRasterDataProvider 对象时，传递到方法中的 TableDescHelper 将为 RasterTableDescHelper 或其子类。此外， TableInfo 也应为 RasterTableInfo 或子类。

向 AddLayerWizard 添加定制 DataProvider

对于任意 DataProvider 的支持均可轻松添加到 AddLayerWizard。该向导可以根据 addlayerwizard.properties 文件中的设置自行初始化。此文件定义：

- 要添加的图层所源自的 DataProvider 列表。
- 对于每个 DataProvider，将提示用户添加 DataProvider 图层所需必要信息的一个或多个 GUI 页面的列表。

- GUI 页面中的某一或全部控件的默认值 （可选）。
- GUI 页面中的某一或全部控件的常用值 （可选）。

AddLayerPage 类

第一步是确定查询构建添加 DataProvider 图层所需的 TableDescHelper 和 DataProviderHelper 所需信息而必须提供的 GUI 页面集。这些页面中的每个页面均将扩展 com.mapinfo.beans.addlayer.AddLayerPage 类。每个 AddLayerPage 本质上均为 javax.swing.JPanel，包括用于该页的所有 GUI 控件。

页类应该调用默认 （无参数） AddLayerPage 构造器，随后将进行向其布局添加 GUI 控件所需的必要工作。

此外，对于需要额外实施的 AddLayerPage，也提供了一些可用的方法。如下所示：

- boolean **validatePage(AddLayerWizard wizard)**: 在向导中单击 “下一步” 或 “完成” 时调用此方法。这也正是验证已经为该页提供所有必需信息的机会。如果一切正常将返回 ture，继续进行后续步骤；如果有问题则返回 false。（此方法的默认实施如果没有覆盖，则会返回 true。）
- void **finishInitializing(AddLayerWizard wizard)**: 在此页变为可见之前 （在向导中此页的前一页单击 “下一步”），将会立即调用此方法。这是执行最后的初始化的机会。（此方法的默认实施如果没有覆盖，则不会作出任何操作。）
- void **setPageLinks(AddLayerWizard wizard)**: 此方法在该页得到成功验证 （即 validatePage() 返回 true）之后将会调用。这为向导中的 “下一页” 或 “上一页” 提供了动态更改机制。（此方法的默认实施如果没有覆盖，则不会作出任何操作。）

有关详细信息，请参阅 Javadocs 中的 com.mapinfo.beans.addlayer.AddLayerPage 类。

AddLayerWizard 类

com.mapinfo.beans.addlayer.AddLayerWizard 类具有某些方法，在页面中完成类似于设置当前 TableDescHelper 和 DataProviderHelper 的操作时将需要利用这些方法。

注： 以上讨论的三个 AddLayerPage 回调方法为您提供了 AddLayerWizard。

这些方法如下所示：

- **setDataProviderHelper(DataProviderHelper dpHelper)**：设置当前的 DataProviderHelper。
- **setTableDescHelper(TableDescHelper tdHelper)**：设置当前的 TableDescHelper。
- **setLayerNameRequired(boolean bNameRequired)**：设置在添加用于当前 DataProvider 的图层时，是否需要图层名。如果使用值 ture 来调用此方法，则将强制用户在向导的最后一步中指定图层名称。

您最可能从获取必要信息的 AddLayerPage(s) 的 validatePage(AddLayerWizard) 方法中调用这些方法，以构建 DataProviderHelper 和 / 或 TableDescHelper。

有关详细信息，请参阅 Javadocs 中的 com.mapinfo.beans.addlayer.AddLayerWizard 类。

addlayerwizard.properties

在实施用于 DataProvider 的必要的页之后，需要添加用于 DataProvider 的条目。为此，需要将新的 DataSourceI 条目添加到 addlayerwizard.properties 中的现有 DataSource 条目集中。有关示例，请参阅 addlayerwizard.properties。

下表说明了支持的键值对：

键	值	必需
DataSourceI	将出现在向导第一步中的数据源名称列表中的数据源名称。	是
DataSourceI_TDHelper_Class	用于此数据源的 TableDescHelper 类的完全类规格。	是
DataSourceI_Allowed_In_Applet	True/False 说明是否允许从 applet 上下文中的这一数据源添加图层 （如未指定则默认为 true）。	否
DataSourceI_Page_Count	此数据源所必需的页数。	是
DataSourceI_PageJ	表示此页的 Java 类。	是
DataSourceI_PageJ_Description	说明此页将要提示用户的具体内容。	是

此外，还可以通过 addlayerwizard.properties 指定 GUI 页面中某些或全部控件的默认值。在此，可以通过 java.awt.Component 的 setName(String) 方法来命名页面中的控件。然后在键中使用相同的名称来定义控件的默认值。

该向导还可以自动将最常用 (MRU) 值用于 addlayerwizard.properties 的命名控件。此外，还可以命名要存储 MRU 值的控件。

19

将 TAB 数据上载到远程数据库

本章介绍如何使用 EasyLoader 将 TAB 数据上载到远程数据库。

本章内容：

EasyLoader 简介

EasyLoader v 7.0 是一个仅适用于 Windows 的实用程序，可从 MapInfo 公司获得，用于将 MapInfo .tab 文件上载到远程数据库，如 MapInfo Professional for Oracle、 Informix 或 SQL Server。在 MapInfo Professional 的安装过程中， EasyLoader 将安装在 Tools 目录下。此外，该程序也可从 MapInfo 公司的网站上免费下载。

支持以下数据库：

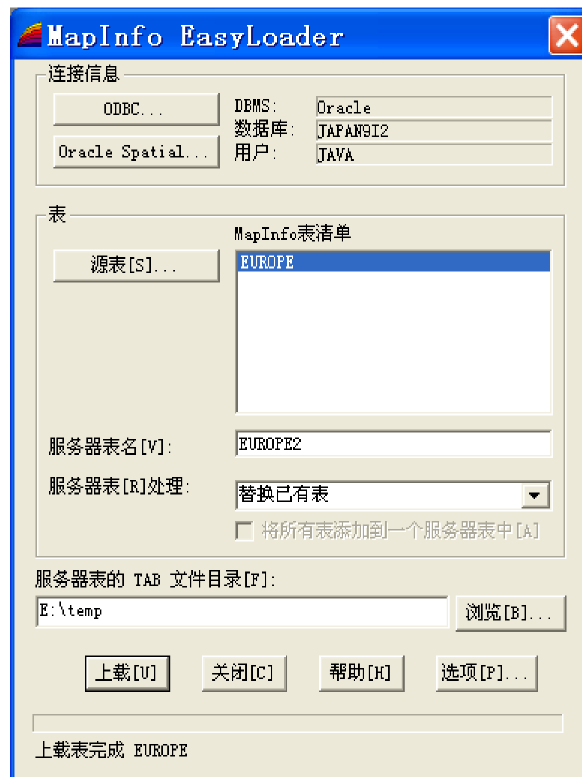
- Oracle Spatial
- Informix Dynamic Server
- SQL Server

对于空间数据库支持， DBMS 需要由自身（如 Oracle Spatial）或借助于附加的 SpatialWare Datablade/Extensions （如 MapInfo SpatialWare for IUS 和 SQL Server）来处理空间几何信息。要支持空间数据库， IUS 驱动程序必须为 2.8 或更高版本。如果上述 DBMS 没有空间对象类型支持，则表格只能作为纬度和经度数据 （XY 或带有 MapInfo 键的 XY (MICODE)）上载。 （任何时候只能打开一个服务器连接。）

运行 EasyLoader

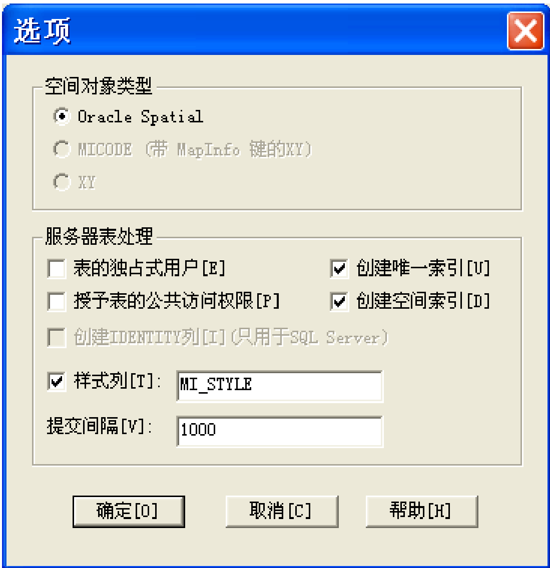
使用 EasyLoader 上载 MapInfo .TAB 文件：

1. 从 MapInfo Professional 的 “工具”菜单运行 EasyLoader。必要时，运行工具管理器以便将 EasyLoader 加载和 / 或自动加载到 “工具”菜单。显示主 EasyLoader 对话框。



2. 在 “连接信息” 下，单击适当的按钮 （ODBC 或 Oracle Spatial）以连接数据库。提供必要的连接信息 （如数据源名称或用户标识、口令和服务器名）。单击确定以返回 EasyLoader 对话框。
3. 单击源表按钮，从单个目录显示 MapInfo 表的列表。当选择了该表进行上载时，名称将显示在 MapInfo 表清单中。
4. 选择表并选择适当的服务器表处理任务 （创建新表、附加到现有表、替换现有表）。在第 322 页的 *EasyLoader 的选项*中介绍了这些选项和附加选项。

   注： 直到选择表之后，“上载” 按钮才可用。
5. 要创建本地 TAB 文件，请提供目录或浏览到其位置。默认情况下， EasyLoader 不生成这些文件。这些表的文件命名约定为 *yourServerTableName*_srv.tab。
6. 要设置上载过程的选项，请单击选项按钮。显示 “选项” 对话框。有关可用选项的说明，请参阅第 322 页的 *EasyLoader 的选项*。



7. 单击上载按钮启动上载过程。当完成上载过程之后，关闭 EasyLoader。
   如果在上载过程中还没有创建 "空间索引"，请在此时通过执行创建索引语句或重新上载表来创建，确保这一次创建 "空间索引" 并替换表 （请参阅步骤 1–3）。

EasyLoader 的选项

表处理选项 （主对话框）

本节介绍表处理选项的主对话框。

创建新表

将用您指定的名称创建服务器表。如果选择了该选项，但服务器上已经有相同名称的一个表，系统会显示一则错误消息通知您。为此需要使用不同的名称或选择选项：替换现有表，以便上载该表。

替换现有表

如果已经存在相同名称的表，那么选择该选项后，该表会被丢弃，将创建一个新的表来匹配上载的 MapInfo 表。

附加到现有表

如果存在服务器表并且两个表的结构相匹配，则 MapInfo 表将被附加到服务器表上。否则，就会出现错误，无法上载该表。表*必须*有相同的表结构，并且处于相同的 Oracle Spatial 投影中。

将所有表附加到一个表

列出的所有 MapInfo 表都将上载到一个服务器表中。该服务器表的名称就是 “服务器表” 框中显示的名称。该图元主要用于将具有相同结构和投影的表上载到一个表中。例如，当选中 “将所有表附加到一个表” 框后，就不会为每个街道图层创建一个新表，而是只创建一个表。这样，所有表都会附加到这个表上。

空间对象类型

从 MapInfo SpatialWare 或 Oracle Spatial（取决于连接的类型）、MICODE（带有键的 XY）和 XY 中进行选择。加载空间数据的默认选项为 MapInfo SpatialWare 或 Oracle Spatial。

MapInfo SpatialWare/Oracle Spatial

要选择该选项，服务器必须为 Oracle 或已经安装了 MapInfo SpatialWare。表将作为空间数据而上载。

XY 和带有 MapInfo 键的 XY (MICODE)

如果服务器不是 Oracle Spatial 或未安装 MapInfo SpatialWare，则使用该选项。数据在服务器上存储为 xy 坐标。因此，服务器表将会创建为点表。如果要上载的 MapInfo 表不是点表，并且已选择该选项，那么您若命令其使用该选项，则将提取其中心并存储在服务器表上。XY 和 MICODE 的不同之处在于 MICODE 将提供 MapInfo 键作为空间索引，因此其性能优于 XY。

表处理选项 （选项对话框）

本节介绍表处理选项的选项对话框。

授权公众访问表

授权公众完全访问服务器表。

专用表

如果已知您将是唯一尝试更新 / 上载表的用户，则可明显缩短大型表的加载时间。如果不选择该选项，则加载程序将在每次提交之后，验证在上载期间该表是否有其他更新。选择该选项之后，就不会执行这种检查，因此可显著加速大型表的运行时。

创建唯一索引

唯一索引创建在 SpatialWare 的 `sw_member` 列， Oracle 的 mi_prinx 列，或 XY 和 MICODE 的 mi_sql_rec_num 列。这些列中是加载程序生成的序列号。这些列总是会产生，但不一定对其进行索引。

创建空间索引

对于 SpatialWare 表，索引创建在名为 sw_geometry 的几何列上。系统将创建一个空间索引，并且在创建 r-tree 索引之后执行 "更新统计"。您还可以构建自己的空间索引以满足特定需求。如果选择这么做，则清除该复选框，以便在加载时节省时间。

对于 Oracle Spatial 表，空间索引创建在几何列上 （默认情况下为 GEOLOC），称为 `<table_name>_SX`。当建立索引之后，将在索引表上运行 ANALYZE 表函数。

创建 IDENTITY 列 （仅适用于 SQL Server）

如果要使用 IDENTITY 属性创建关键字列 (sw_member)，则选中该复选框。当使用此功能时， SQL Server 将自动生成唯一的关键字列值。当插入新行时，无需手动填写关键字。

在 7.0 版中，默认情况下关键字列 (sw_member) 是用 IDENTITY 属性创建的，其行为与先前版本 (6.8) 的行为相反。若要在没有 IDENTITY 的情况下上载表，请选择 "选项" 按钮，清除 "创建 IDENTITY 列"。

样式列

指定是否随数据加载每行的符号体系。在指定的列中，符号体系加载为一个文本字符串。可以在编辑文本框中指定要使用的列名称。此名称被初始化为默认的列名称，即 MI_STYLE。

注： 要加载每行的符号体系，数据库的 MAPINFO_MAPCATALOG 必须包含以下列： RENDITIONTYPE、 RENDITIONCOLUMN、 RENDITIONTABLE 和 NUMBER_ROWS。有关详细信息，请参阅*附录 C: MAPINFO.MAPINFO_ MAPCATALOG* 。

加载 Oracle Spatial 数据

以下内容介绍如何加载 Oracle 空间数据。

将纬度 / 经度表加载到 Oracle 9i 中

在加载使用了纬度 / 经度坐标系 （测地线数据）的表到 Oracle 9i 中时，验证所有几何坐标是否在经度 (-180,180) 和纬度 (-90, 90) 之间非常重要。Oracle Spatial 不支持超出此范围的测量数据坐标，这些坐标可能产生问题。您可以在加载前使用 MapInfo Professional 检查数据，也可以在加载到 Oracle Spatial 后使用表上的 Oracle Spatial SDO_GEOM.VALIDATE_LAYER( ) 函数检查数据。

注： 当加载 Oracle 9i 时， EasyLoader 将默认的基准 1984 年全球测量坐标系 /WGS84 指定为无基准的纬度 / 经度坐标系。

验证 Oracle 数据

在此可使用两个函数验证 Oracle 上的数据：

- SDO_GEOM.VALIDATE_GEOMETRY( )
- SDO_GEOM.VALIDATE_LAYER( )

这些函数仅仅因为 EasyLoader 所设置的容错度级别，就可能产生验证错误，比如：裲 RA-13356 几何中的相邻点冗余，或 裲 RA-13022 多边形自交。USER_SDO-GEOM_METADATA 中的容错度可向下调节 （以 10 为因子）并重新进行验证。但如果调整了容错度，则必须重新创建空间索引，因为它们使用的是创建时的容错度。

命令行标记

Easyloader 支持以下命令行。

- **/A** 将所有表附加到一个表
  只要表结构相同，此标记允许将多个表上载到一个表。
  语法：/A
- **/D** 服务器创建表
  当您提供 TAB 文件目录时，生成 TAB 文件以访问远程 DBMS。默认情况下，EasyLoader 不生成这些文件。新生成的 TAB 文件为 “服务器表名称” 加上 _srv.TAB。

此目录必须是上载表的有效目录。空目录视为有效。

命令行选项为 /D 路径名

语法：/D "C:\tabfiles"

- **/E** 专用表

  如果已知您将是唯一尝试更新表的人，则可明显缩短大型表的加载时间。指定此标记并不能保证加载程序实现专用；您必须针对加载程序保证专用。

  加载程序在每次提交之后检查主键列的当前最大值，以确保它检测到其他过程可能执行的其他任何输入。此标记可阻止这种检查的发生，因此可显著加速大型表的运行时。

  此标记可放置在快捷方式中，以便交互使用 EasyLoader 界面来执行其他功能。

  语法：/E

- **/F** 日志文件名

  日志文件始终会生成，此标记可以使用户指定日志文件的名称和位置。默认名称为 EasyLoader.log，将在用户的临时目录内创建。如果使用日志文件标记，但没有指定路径，则日志将在 EasyLoader.exe 文件所在的目录中生成。

  语法：第一个示例表明只指定了日志文件的名称，这种情况下日志将在 EasyLoader.exe 所在的目录中生成；第二个示例指定了日志文件的完整路径。

  /F myLogFile.txt

  /F c:\temp\myLogFile.txt

- **/G** 全部授权

  此标记将对公众授予所有权限。在默认情况下，此标记为关闭状态。

  语法：/G

- **/I** 不创建空间索引

  在默认情况下，此标记为关闭状态。当关闭此标记时，将创建空间索引。当打开此标记时，不能在表上创建空间索引。

  对于 Informix Dynamic Server，将创建空间索引，然后发出 “更新统计媒介”语句。请参阅 /U 标记，该标记控制唯一索引。

  对于 Oracle Spatial 表，空间索引创建在几何列上，称为 <table_name>_SX。

  对于 SpatialWare 表，索引创建在 column 几何列上，称为 hg<table_name>ind。

  语法：/I

- **/K** 创建自动关键字列，仅适用于 **SQL Server**

  作为一个选项，关键字列 (sw_member) 可用 IDENTITY 属性来创建。当使用此功能时， SQL Server 将自动生成唯一的关键字列值，当插入新行时，用户不需要手动填写关键字。

在 EasyLoader 7.0 版中，默认情况下关键字列 (sw_member) 是用 IDENTITY 属性创建的，它与先前的版本 (6.8) 相反。因此，在命令行中不指定 K 选项与指定为 /K 时的操作相同，即用 IDENTITY 属性创建关键字列。如果您要关闭该属性，则需要在 K 后提供关键字 “NO_IDENTITY”。

示例：/K NO_IDENTITY

语法：/K

- **/L** 列出 **MapInfo** 表
  此标记允许用户指定一个文本文件，其中列出要上载的表。每一行的格式与 /T 标记相同。

  语法：/LListOTables.txt

- **/M MICODE/XY ?**
  此标记允许用户指定在使用 SpatialWare 的情况下，要使用的对象类型。如果使用 /M 标记，则用户必须在 /M 后提供 MICODE （对于带有 MapInfo 键的 XY）或 XY （对于 XY）。在 /M 后出现 MICODE 或 XY 以外的任何单词都被视为错误。如果不使用 /M 标记，则当所选的数据库已安装 SpatialWare 时，使用 SpatialWare 作为默认选项。

  语法：/M micode

  /M xy

- **/O** 连接字符串
  此标记允许将 Oracle Spatial 的连接字符串传递到程序。有关 ODBC 连接的信息，请参阅 /S 标记。

  语法：/O user_name/password@server_name

- **/P ACR**
  此标记指定与加载到服务器的表的关系。您可指定三个选项之一：’A’、’C’ 或 ’R’。选择 ’A’ 将附加现有的服务器表；选择 ’C’ 将创建新的服务器表；选择 ’R’ 将替换现有表。不能指定多个选项。如果服务器上存在相同名称的表，创建表标记将会失败。

  语法：/P A

  注：　　此标记只是为获得向后兼容性。

- **/Q** 退出
  此标记强制加载程序在完成后退出。

  语法：/Q

- **/R** 替换服务器表

此标记将导致丢弃服务器表。创建并上载新表。默认情况下，它将创建在服务器表上。

语法：/R/

- **/T MapInfo** 表名称 **;** 服务器表名称 **;** 范围

  此标记允许将单个表以不同的名称上载，并限制上载的记录数。MapInfo 表名称、服务器名称和范围之间的分隔符为分号。范围的格式为：起始记录 < 逗号 > 结束记录。服务器表名称和范围是可选的。

  语法：/T c:\data\us_cnty.tab;counties;1,500

- **/U** 不创建唯一索引

  在默认情况下，此标记为关闭状态。当关闭此标记时，表上将创建唯一索引。当打开此标记时，不能在表上创建唯一索引。请参阅 /I 标记，该标记控制空间索引。对于 Oracle Spatial 表，唯一索引创建在 MI_PRINX 列上，称为 <table_name>_IDX，对于 SpatialWare 表，该索引创建在 SW_MEMBER 列上，称为 <table_name>_i。

  语法：/U

- **/V Oracle** 版本

  此标记可用于在 Oracle 8.1.6 服务器上用 8.1.5 格式加载表。通常不建议这么做，但如有特殊需要，也可采用。

  如果要与对话框一起使用此标记，则请参阅下一页的 “混合命令行标记与 GUI” 一节。

  语法：/V

- **/X** 提交时间间隔

  此标记用户用户指定提交时间间隔。当到达提交时间间隔时， Easyloader 将提交插入的报告。默认的提交时间间隔为 1000。创建 Oracle Spatial 的空间索引时也使用与此相同的时间间隔。如果提交时间间隔设置为 0 （零），则整个范围的记录将作为一个事务在执行提交之前插入。

  语法：/X 500

- **/Y** 符号体系列名称

  此标记可用于指定将与数据一起加载每行的符号体系。此外还可指定要使用的列的名称。如果未提供任何名称，则使用默认的列名称，即 MI_STYLE。

  语法：/Y *StyleColumnName*

混合命令行标记与 GUI

使用 Windows 快捷方式，可将命令行标记与 EasyLoader 界面混合起来。这样就更容易将标记设置为默认标记，同时也能够从 UI 将其覆盖。有些标记只能从命令行使用。

- 创建 EasyLoader 的快捷方式。
- 右键单击此快捷方式，并选择 “属性”。
- 在 “快捷方式”选项卡下，在 “目标”编辑框内，在该行的最后添加命令行标记。

当通过快捷方式运行 EasyLoader 时，指定的标记就会生效。

20

Web 地图服务

MapXtreme Java 提供 Web 地图服务 (WMS) 供用户使用。MapXtreme Java 中的 WMS 兼容 1.1.1 OpenGIS® 的 Web 地图服务实施规范。本文档也可参阅 **http://www.opengis.org/techno/implementation.htm**。

本章内容：

WMS 服务器概览

MapXtreme Java 中的 WMS 符合 Open GIS 协会的 Web 地图服务实施规范，可用于实现以下操作：

- GetCapabilities — 返回服务级别的元数据。这一元数据是有关信息内容和服务的可接受请求参数的说明。
- GetMap — 返回已定义地理空间和尺寸参数的地图图像。
- GetFeatureInfo — 返回有关地图所示图元的信息。此操作为可选。

注： 所有这些请求类型均为区分大小写。

GetCapabilities

GetCapabilities 返回服务级别的元数据。这一元数据是有关信息内容和服务的可接受请求参数的说明。

请求参数

下表列出了可能的请求参数。

请求参数	必需	说明
VERSION=version	否	请求版本。MapXtreme Java 只支持 1.1.1 版本。
SERVICE=WMS	是	服务类型。由于目前所有 MapXtreme Java 均支持 WMS，因此 SERVICE 属性始终设置为 WMS。
REQUEST=GetCapabilities	是	请求名称

示例请求

以下是 GetCapabilities 请求的示例：

```
http://hostname:portnumber/wmsserver111/servlet/wms?SERVICE=WMS
&REQUEST=GetCapabilities
```

示例响应

以下是 GetCapabilities 响应的示例：

```
- <WMT_MS_Capabilities version="1.1.1">
  - <Service>
      <Name>OGC:WMS</Name>
      <Title>MapInfo Corporation Map Server</Title>
      <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
      xlink:type="simple" xlink:href="http://hostname:portnumber/
      wmsserver111/servlet/wms?" />
    </Service>
  - <Capability>
    - <Request>
      - <GetCapabilities>
        <Format>application/vnd.ogc.wms_xml</Format>
       - <DCPType>
        - <HTTP>
          - <Get>
              <OnlineResource xmlns:xlink="http://www.w3.org/
              1999/xlink" xlink:type="simple" xlink:href="http:/
              /hostname:portnumber/wmsserver111/servlet/wms?" />
            </Get>
          - <Post>
              <OnlineResource xmlns:xlink="http://www.w3.org/
              1999/xlink" xlink:type="simple" xlink:href="http:/
              /hostname:portnumber/wmsserver111/servlet/wms?" />
            </Post>
          </HTTP>
         </DCPType>
        </GetCapabilities>
      - <GetMap>
         <Format>image/gif</Format>
         <Format>image/jpeg</Format>
         <Format>image/jpg</Format>
         <Format>image/png</Format>
       - <DCPType>
        - <HTTP>
          - <Get>
              <OnlineResource xmlns:xlink="http://
              www.w3.org/1999/xlink" xlink:type="simple"
              xlink:href="http://hostname:portnumber/
              wmsserver111/servlet/wms?" />
            </Get>
          - <Post>
              <OnlineResource xmlns:xlink="http://
              www.w3.org/1999/xlink" xlink:type="simple"
              xlink:href="http://hostname:portnumber/
              wmsserver111/servlet/wms?" />
```

```
         </Post>
       </HTTP>
      </DCPType>
     </GetMap>
   - <GetFeatureInfo>
       <Format>text/xml</Format>
    - <DCPType>
     - <HTTP>
       - <Get>
           <OnlineResource xmlns:xlink="http://www.w3.org/
           1999/xlink" xlink:type="simple" xlink:href="http:/
           /hostname:portnumber/wmsserver111/servlet/wms?" />
         </Get>
       - <Post>
           <OnlineResource xmlns:xlink="http://www.w3.org/
           1999/xlink" xlink:type="simple" xlink:href="http:/
           /hostname:portnumber/wmsserver111/servlet/wms?" />
         </Post>
       </HTTP>
      </DCPType>
     </GetFeatureInfo>
   </Request>
 - <Exception>
    <Format>application/vnd.ogc.se_xml</Format>
    <Format>application/vnd.ogc.se_inimage</Format>
    <Format>application/vnd.ogc.se_blank</Format>
   </Exception>
 - <Layer queryable="0" opaque="0" noSubsets="0">
    <Title>MapInfo Corporation Map Server</Title>
    <SRS>EPSG:4201</SRS>
    <SRS>EPSG:4205</SRS>
    [...]
    <LatLonBoundingBox maxx="180" miny="-90" minx="-180"
    maxy="90" />
    - <Style>
       <Name>mistyles/brushes/brush_001</Name>
       <Title>brush_001</Title>
       </Style>
       [...]
   </Layer>
 - <Layer queryable="1" opaque="0" noSubsets="0">
       <Name>Layers/World/Capitals</Name>
       <Title>World Capitals</Title>
       <SRS />
       <LatLonBoundingBox maxx="0.0038672423472374423"
       miny="0.001097654483307838" minx="-9.318052647195836E-4"
       maxy="0.0021815925889656857" />
   </Layer>
 - <Layer queryable="1" opaque="0" noSubsets="0">
```

```
            <Name>Layers/World/Countries</Name>
            <Title>World Countries</Title>
            <SRS />
            <LatLonBoundingBox maxx="0.00497406046898038"
            miny="7.405105032985079E-4" minx="-0.001959711056303122"
            maxy="0.0023619945565068867" />
        </Layer>
      - <Layer queryable="1" opaque="0" noSubsets="0">
            <Name>Layers/World/Grid</Name>
            <Title>Grid</Title>
            <SRS />
            <LatLonBoundingBox maxx="0.005025125685412717"
            miny="7.405105008545692E-4" minx="-0.0021343026367284536"
            maxy="0.002421524944647251" />
        </Layer>
      - <Layer queryable="1" opaque="0" noSubsets="0">
            <Name>Layers/World/Ocean</Name>
            <Title>Ocean (Robinson)</Title>
            <SRS />
            <LatLonBoundingBox maxx="0.005025098523972836"
            miny="7.405105008635173E-4" minx="-0.0021343326671474454"
            maxy="0.0024215249446383038" />
        </Layer>
      </Capability>
    </WMT_MS_Capabilities>
```

存储服务元素信息

用户可以提供一个 service.xml 文件，该文件存储了符合 OGC 定义的服务元素信息。 这一 service.xml 文件需要符合 OGC DTD，并且需要位于 WEB-INF 文件夹的 wmsserver111 上下文中。以下提供了 service.xml 文件的示例。XML 文件的根元素需要为 <Service>，否则系统会将 XML 文件视为无效，并且 getCapabilitiesRequest 将出现故障。

```
- <Service>
  <Name>OGC:WMS</Name>
  <Title>Custom Service title</Title>
  <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
    xlink:type="simple" xlink:href="http://198.242.23.138:8080
    /wmsserver111/servlet/wms?" />
  </Service>
```

GetCapabilities 请求高速缓存

MapXtreme Java 在初始化时创建了一个 capabilities.xml 文件，用于在 MapXtreme Java 对 GetCapabilities 请求作出响应时进行高速缓存。

此文件可以修改，以便提供有关数据的更多自定义信息。此外，还有一个新的初始化参数用于控制此文件的 “生存期”。以下提供了与其相关的 web.xml 的示例：

```
<!--
    A value of -1 means that the capabilities.xml file never expires.
    A positive value is interpreted as the maximum allowed age of the
    file in hours.  When the server is initialized, it will check the
    age of the file, and if it has become outdated, it will regenerate
    a new version of the file.
-->
    <init-param>
        <param-name>maxAgeOfCapabilitiesXML</param-name>
        <param-value>-1</param-value>
    </init-param>
```

GetMap

GetMap 返回已定义地理空间和尺寸参数的地图图像。在调用 GetMap 时， WMS 客户端将指定：

- 图层
- 样式
- 边界框
- 参考坐标系
- 输出格式
- 输出大小
- 背景透明度和颜色

请求参数

下表列出了可能的请求参数。

请求参数	必需	说明
VERSION=version	是	请求版本。MapXtreme Java 只支持 1.1.1 版本。
REQUEST=GetMap	是	请求名称。
LAYERS=layer_list	是	由逗号间隔的列表，其中列出了一个或多个地图图层。
STYLES=style_list	是	由逗号间隔的列表，其中列出了渲染样式每个请求的图层。
SRS=namespace:identifier	是	空间参考系统。
BBOX=minx,miny,maxx,maxy	是	采用 SRS 单位表示的边界框角 （左下、右上）。
WIDTH=output_width	是	地图图片的像素宽度。
HEIGHT=output_height	是	地图图片的像素高度。
FORMAT=output_format	是	地图的输出格式。
TRANSPARENT =TRUE|FALSE	否	地图的背景透明度 （默认为 FALSE）。
BGCOLOR=color_value	否	背景色的十六进制红绿蓝颜色值 （默认值为 0xFFFFFF）。
EXCEPTIONS =exception_format	否	WMS 的异常错误报告格式 （默认为 SE_XML）。
WFS =web_feature_service_URL	否	提供要使用样式图层说明器符号化的图元的 Web 图元服务的 URL。

示例请求

以下是 GetMap 请求的示例：

```
http://hostname:portnumber/wmsserver111/servlet/wms?VERSION=1.1.1
&SRS=epsg:4267&REQUEST=GetMap&LAYERS=Layers/World/
Countries&STYLES=&BBOX=-180,-180,180,180
&WIDTH=800&HEIGHT=600&FORMAT=image/gif
```

示例响应

以下是 GetMap 响应的示例：



GetFeatureInfo

GetFeatureInfo 返回有关地图所示图元的信息。此操作为可选。如果采用了 GetFeatureInfo，则必须在 GetMap 请求之后发出。

请求参数

下表列出了可能的请求参数。

请求参数	必需	说明
VERSION=version	是	请求版本。MapXtreme Java 只支持 1.1.1 版本。
REQUEST=GetFeatureInfo	是	请求名称。
<map_request_copy>	是	生成用于所需信息的地图的地图请求参数的副本。
QUERY_LAYERS=layer_list	是	逗号分隔的列表，其中列出了一个或多个要查询的图层。
INFO_FORMAT =output_format	否	图元信息的返回格式（MIME 类型）。
FEATURE_COUNT=number	否	要返回有关其信息的图元数量（默认为 1）。

请求参数	必需	说明
X=pixel_column	是	以像素表示的图元 X 坐标 （自左上角测量为 0）
Y=pixel_row	是	以像素表示的图元 Y 坐标 （自左上角测量为 0）
EXCEPTIONS =exception_format	否	WMS 的异常错误报告格式 （默认为 application/ vnd.ogc.se_xml）。

示例请求

以下是 GetFeatureInfo 请求的示例：

```
http://hostname:portnumber/wmsserver111/servlet/wms?VERSION=1.1.1
&REQUEST=GetFeatureInfo&SRS=epsg:4326&LAYERS=Layers/World/Countries
&STYLES=&BBOX=-180,-180,180,180&WIDTH=800&HEIGHT=600&QUERY_LAYERS=Layers/
World/Countries&X=54&Y=54
```

示例响应

以下是 GetFeatureInfo 响应的示例：

```
- <AnnFeatureSetList>
  - <AnnFeatureSet>
      <name>world</name>
    - <boundedBy>
      - <Box srsName="mapinfo:coordsys122">
       - <coord>
           <X>-1.6194966287191512E7</X>
           <Y>-8621185.324024437</Y>
         </coord>
       - <coord>
           <X>1.6789976633244906E7</X>
           <Y>8326222.646170927</Y>
         </coord>
        </Box>
      </boundedBy>
    - <CoordinateReferenceSystem>
      - <Identifier>
         <code>coordsys122</code>
         <codeSpace>mapinfo</codeSpace>
        </Identifier>
      </CoordinateReferenceSystem>
    - <AttNameTypedTuple>
        <AttNameTyped Type="string">Country</AttNameTyped>
        <AttNameTyped Type="string">Capital</AttNameTyped>
        <AttNameTyped Type="decimal">Pop_1994</AttNameTyped>
        <AttNameTyped Type="decimal">Pop_Grw_Rt</AttNameTyped>
```

```
        <AttNameTyped Type="decimal">Pop_Male</AttNameTyped>
        <AttNameTyped Type="decimal">Pop_Fem</AttNameTyped>
        <AttNameTyped Type="decimal">Pop_0_14</AttNameTyped>
        <AttNameTyped Type="decimal">Pop_15_64</AttNameTyped>
        <AttNameTyped Type="decimal">Pop_65Plus</AttNameTyped>
        <AttNameTyped Type="decimal">Male_0_14</AttNameTyped>
        <AttNameTyped Type="decimal">Male_15_64</AttNameTyped>
        <AttNameTyped Type="decimal">Male_65Plus</AttNameTyped>
        <AttNameTyped Type="decimal">Fem_0_14</AttNameTyped>
        <AttNameTyped Type="decimal">Fem_15_64</AttNameTyped>
        <AttNameTyped Type="decimal">Fem_65Plus</AttNameTyped>
        <AttNameTyped Type="decimal">Pop_Urban</AttNameTyped>
        <AttNameTyped Type="decimal">Pop_Rural</AttNameTyped>
        <AttNameTyped Type="decimal">Pop_Urb_Male</AttNameTyped>
        <AttNameTyped Type="decimal">Pop_Urb_Fem</AttNameTyped>
        <AttNameTyped Type="decimal">Pop_Rur_Male</AttNameTyped>
        <AttNameTyped Type="decimal">Pop_Rur_Fem</AttNameTyped>
        <AttNameTyped Type="decimal">Arable_Pct</AttNameTyped>
        <AttNameTyped Type="decimal">Literacy</AttNameTyped>
        <AttNameTyped Type="decimal">Inflat_Rate</AttNameTyped>
        <AttNameTyped Type="decimal">Unempl_Rate</AttNameTyped>
        <AttNameTyped Type="decimal">Indust_Growth</AttNameTyped>
        <AttNameTyped Type="string">Continent</AttNameTyped>
        <AttNameTyped Type="int">MapInfo_ID</AttNameTyped>
      </AttNameTypedTuple>
    - <KeyColumns>
        <AttName>MapInfo_ID</AttName>
      </KeyColumns>
    </AnnFeatureSet>
  </AnnFeatureSetList>
```


WMS 栅格数据提供方

Mapxtreme Java 提供了 WMSRasterDataProviderHelper 和 WMSRasterTableDescHelper 类，可用于使用任意 WMS v1.1.1 Web 服务，向 MapXtreme Java 提供将要作为地图组成部分的图层。WMS 栅格数据提供方只可用于渲染图层。和渲染无关的任意查询或者返回空 FeatureSet 或者返回没有意义的结果。

注： 返回有效结果的唯一一种查询是需要渲染的查询。有效的结果将返回自 queryInRectangle。

类 WMSRasterDataProviderHelper 说明要使用的 WMS 1.1.1 服务器的位置。该类和 WMSRasterTableDescHelper 一起使用，可以向基于 WMS 1.1.1 服务器上的图层的 MapJ 对象添加图层。MapXtreme Java 目前不允许远程访问栅格图层。由于此数据提供方提供栅格信息，因此其不能通过使用 MapXtremeDataProviderRef 进行远程数据访问。

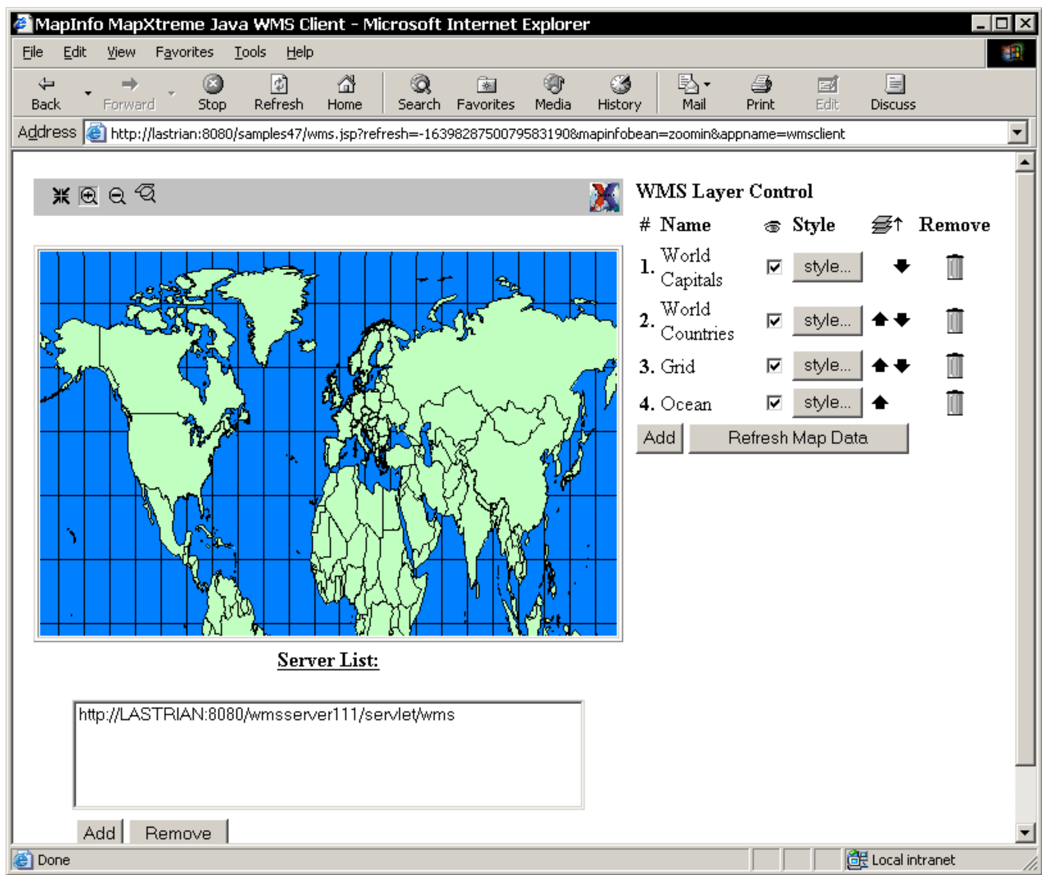
类 WMSRasterTableDescHelper 和 WMSRasterDataProviderHelper 一起使用，可用于说明使用 WMS 1.1.1 数据源的 MapJ 对象中的图层。

有关详细信息，请参阅 Javadocs 中有关相应的类的说明。

WMS 客户端

MapXtreme Java 版提供了 JSP WMS 定制标记示例应用程序用于显示 WMS 客户端的示例功能。打开浏览器，键入指向示例的 URL，即可访问 JSP WMS 定制标记，该 URL 可如下所示：

```
http://localhost:8080/samples47/htmlmap
```



WMS 客户端界面说明

本节介绍 WMS 客户端的图形用户界面。

服务器列表

在该界面的服务器列表部分提供了以下项目。

Add — 向列表添加新服务器，并令其图层可用于添加图层控制。此时将忽略无效的 WMS URL。

Remove — 从服务器列表删除服务器，并从地图删除其所有图层。

地图工具

在该界面的地图工具部分提供了以下项目。

ZoomIn/ZoomOut/Recenter — 与地图一起使用的基本地图工具。

ZoomToAllLayers — 更改缩放以包含地图上的所有可见图层。

图层控制

在该界面的图层控制部分提供了以下项目。

Visibility （开 / 闭）— 更改地图图层的可视性。

Style Chooser — 用于从图层支持的样式的列表中选择样式。

Move Layer （上 / 下）— 更改地图图层的排序顺序。

Remove Layer — 增减随任意可用服务器提供的图层。

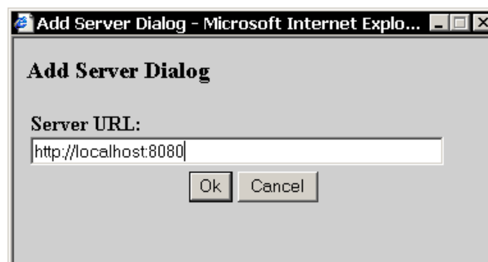
Add Layer — 添加已经随任意可用服务器提供的图层。

Refresh Map Data — 考虑任意图层可见性的更改，重画图层。

使用 WMS 客户端

要使用 WMS 客户端，可执行以下操作：

1. 从界面的 Server List 部分，单击 **ADD**。此时将显示 “Add Server Dialog ”框。



2. 输入所要使用的 WMS 1.1.1 服务器的 URL，然后单击 **OK**。
3. 从界面的 WMS 图层控制部分，单击 **ADD**。此时将显示 “Add Layer”对话框。



4. 从相应的下拉框中选择服务器、图层和图像类型，然后单击 **OK**。
5. 使用第 342 页的 *WMS 客户端界面说明* 中介绍的控件对地图进行所需的操作。

第 4 部分：附录

本开发指南的 第 4 部分提供了众多参考资料，用于帮助您进一步充分利用 MapXtreme Java。

主题：



定制 JSP 标记库

本附录提供了定制 JSP 标记的列表，并且说明了如何自行创建定制标记并将其展示在 MapXtreme Java 管理器的 Web 应用程序构建器中。

本附录内容：

定制 JSP 标记

以下是可用于 MapXtreme Java 的定制 JSP 标记。

元素	说明	父级	**JSP** 语法
mapapp	MapInfo 所有 JSP 定制标记的根标记。定义所有生成的图像的应用程序名称和 mime 类型。		<mapinfo:mapapp [name="appname"] [mimeType="mime"]>…</ mapinfo:mapapp> **web.xml** 中 **RequestHandler** 的 **servlet** 名称应该和 **mapinfo:mapapp** 标记中指定的 **appname** 相同。
toolbar	工具栏工具的根标记。	mapapp	<mapinfo:toolbar>…</ mapinfo:toolbar>
tool	工具栏的一般工具。	toolbar	<mapinfo:tool page="relativeURL" name="imgName" img="relativeURL" [options="JavaScript window Options"] [alt="text"] [width="size"] [height="size"]
colors	显示包含供选颜色的 HTML 选择框。其中每个元素的值均为颜色的 int 值。	mapapp	<mapinfo:colors name="formName" [selected="selectedColor"] />
cancel	显示用于关闭当前浏览器窗口的 HTML 按钮。如果提供了 src，该按钮将输出为图像。	mapapp	<mapinfo:cancelButton [value="buttonCaption"] [src="relativeURL"] />
map	显示地图。	mapapp	<mapxtreme:map />
resizableMap	显示可以重新调整大小的地图。要令地图大小可以调整，此标记不能嵌入到表中。有关详细信息，请参阅 Javadoc 中的 MapTag。	mapapp	<mapxtreme:resizableMap resizable="true|false" />

<table>
<tr><th>元素</th><th>说明</th><th>父级</th><th>JSP 语法</th></tr>
<tr><td>printPreviewMap</td><td>显示其大小由标记属性确定的地图。此地图只能用于显示。平移、缩放等功能将不能和此标记一起正常工作。有关示例请参阅 printpreview.jsp。</td><td>mapapp</td><td><mapxtreme:printPreviewMap width=”size” height=”size” /></td></tr>
<tr><td>layercontrol</td><td>用于图层控制的根标记。定义图层控制是否在和地图相同的页面上。有关示例请参阅 layercontroldialog.jsp。</td><td>mapapp</td><td><mapxtreme:layercontrol [dialog=”true|false”] >…</ mapxtreme:layercontrol></td></tr>
<tr><td>layerlist</td><td>可见于各个图层和 MapJ 中的 ThemeList。定义在图层控制中要输出的主题。</td><td>layercontrol</td><td><mapxtreme:layerlist [showRangedThemes=”true|false”] [showIndivisualValueThemes=”true|false”] [showOverrideThemes=”true|false”] [showSelectionThemes=”true|false”] >… </mapxtreme:layerlist></td></tr>
<tr><td>visible</td><td>输出 HTML 复选框，用于更改当前图层的可见性。</td><td>layerlist</td><td><mapxtreme:visible /></td></tr>
<tr><td>标注</td><td>输出 HTML 复选框，用于更改当前图层的自动标注。</td><td>layerlist</td><td><mapxtreme:label /></td></tr>
<tr><td>select</td><td>输出 HTML 复选框，用于更改当前图层的可选性。</td><td>layerlist</td><td><mapxtreme:visible /></td></tr>
<tr><td>layerIndex</td><td>输出当前图层的索引，如果当前图层是专题则不输出。</td><td>layerlist</td><td><mapxtreme:layerIndex /></td></tr>
<tr><td>layername</td><td>输出图层名称或专题名称。</td><td>layerlist</td><td><mapxtreme:layername [tableInfo=”true|false”] [page=”true|false”] [options=”JavaScript Window Options”] /></td></tr>
<tr><td>displayOptionsTool</td><td>输出一个工具，用于打开更改当前图层显示选项的对话框。如果包含值，则输出 HTML 按钮，否则将输出图像。</td><td>layerlist</td><td><mapxtreme:displayOptionsTool [value=”text”] [layer=”index”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] [options=”JavaScript Window Options”] [page=”relativeURL”] /></td></tr>
</table>

元素	说明	父级	**JSP** 语法
labelOptionsTool	输出一个工具，用于打开更改当前图层标注选项的对话框。如果包含值，则输出 HTML 按钮，否则输出图像。	layerlist	<mapxtreme:labelOptionsTool [value=”text”] [layer=”index”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] [options=”JavaScript Window Options”] [page=”relativeURL”] />
fontOptionsTool	输出一个工具，用于打开更改当前图层字体选项的对话框。如果包含值，则输出 HTML 按钮，否则输出图像。	layerlist	<mapxtreme:fontOptionsTool [value=”text”] [layer=”index”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] [options=”JavaScript Window Options”] [page=”relativeURL”] />
removeLayer	输出用于删除专题或当前图层的按钮。	layerlist	<mapxtreme:removeLayer [src=”relativeURL”] [width=”size”] [height=”size”] />
layercontroltool	输出用于打开图层控制对话框的工具栏工具。	toolbar	<mapxtreme:layercontroltool [page=”relativeURL”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] />
zoomin	输出用于放大地图的工具栏工具。	toolbar	<mapxtreme:zoomin [alt=”text”] />
zoomout	输出用于缩小地图的工具栏工具。	toolbar	<mapxtreme:zoomout [alt=”text”] />
recenter	输出用于地图重定中心的工具栏工具。	toolbar	<mapxtreme:recenter [alt=”text”] />
infotool	输出一个工具栏，用于在某点执行搜索并显示每个图层在该点的有关信息。	toolbar	<mapxtreme:infotool [page=”relativeURL”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] />
info	可见于每个图层的标记，为每个图层设置 FeatureSet。有关示例请参阅 infodialog.jsp。	mapapp	<mapxtreme:info [searchMode=”SearchMode constant”] >…</mapxtreme:info>
selectiontool	输出用于选择地图对象的工具栏工具。	toolbar	<mapxtreme:selectiontool [searchMode=”SearchMode constant”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] />

元素	说明	父级	**JSP** 语法
unselecttool	输出用于取消选择所有地图对象的工具栏工具。	toolbar	<mapxtreme:unselecttool [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] />
selectioninfotool	输出一个工具栏工具，用于打开显示所选图元属性数据的对话框。	toolbar	<mapxtreme:selectioninfotool [page=”relativeURL”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] />
selectionInfo	可见于每个图层的标记，为每个可选图层设置 FeatureSet。有关示例请参阅 selectioninfodialog.jsp。	mapapp	<mapxtreme:selectionInfo>…</mapxtreme:selectionInfo>
opentool	输出一个工具栏工具，用于启动选择要打开的 MDF 或 geoset 的对话框。	toolbar	<mapxtreme:opentool [page=”relativeURL”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] [options=”JavaScript Options” />
open	输出 HTML 选择框，列出给定目录中的 MDF 和 geoset。有关示例请参阅 opendialog.jsp。	mapapp	<mapxtreme:open [directory=”path”] />
savetool	输出一个工具栏工具，用于打开保存 MDF 的对话框。	toolbar	<mapxtreme:savetool [page=”relativeURL”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] [options=”JavaScript Options” />
save	输出一个 HTML 文本框，用于输入要保存的 MDF 的文件名。有关示例请参阅 savedialog.jsp。	mapapp	<mapxtreme:save [dialog=”true|false”] [directory=”path”] [size=”textbox size”] />
displayOptions	用于显示选项标记的根标记。定义此标记是否在对话框中。有关示例请参阅 displayoptionsdialog.jsp。	mapapp	<mapxtreme:displayOptions [layer=”layerIndex”] [dialog=”true|false”]>…</mapxtreme:displayOptions>
displayZoomRange	输出一个 HTML 复选框，用于更改当前的图层是否显示在缩放范围之内。	displayOptions	<mapxtreme:displayZoomRange />

元素	说明	父级	**JSP** 语法
displayMinZoom	输出一个 HTML 文本框，用于更改当前图层的最小缩放范围。	displayOptions	<mapxtreme:displayMinZoom [size=”size”] />
displayMaxZoom	输出一个 HTML 文本框，用于更改当前图层的最大缩放范围。	displayOptions	<mapxtreme:displayMaxZoom [size=”size”] />
labelOptions	用于标注选项标记的根标记。定义此标记是否在对话框中。有关示例请参阅 labeloptionsdialog.jsp。	mapapp	<mapxtreme:labelOptions [layer=”layerIndex”] [dialog=”true|false”]>…</ mapxtreme:labelOptions>
labelZoomRange	输出一个 HTML 复选框，用于更改是否在缩放范围之内显示当前图层的标注。	labelOptions	<mapxtreme:labelZoomRange />
labelMinZoom	输出一个 HTML 文本框，用于更改标注的最小缩放范围。	labelOptions	<mapxtreme:labelMinZoom [size=”size”] />
labelMaxZoom	输出一个 HTML 文本框，用于更改标注的最大缩放范围。	lableOptions	<mapxtreme:labelMaxZoom [size=”size”] />
labelColumns	输出一个选择框，用于选择将哪一列用于标注。	labelOptions	<mapxtreme:labelColumns />
labelDuplicates	输出一个复选框，用于更改是否显示重复的标注。	labelOptions	<mapxtreme:labelDuplicates />
labelOverlapping	输出一个复选框，用于更改是否输出重叠的标注。	labelOptions	<mapxtreme:labelOverlapping />
fontOptions	用于字体选项标记的根标记。定义此标记是否在对话框中。有关示例请参阅 fontoptionsdialog.jsp。	mapapp	<mapxtreme:fontOptions [layer=”layerIndex”] [dialog=”true|false”]…</ mapxtreme:fontOptions>
fontList	输出一个带有系统所有可用字体的选择框。	fontOptions	<mapxtreme:fontList />
fontSize	输出一个 HTML 文本框，用于更改标注的字体大小。	fontOptions	<mapxtreme:fontSize [size=”size”] / >
fontColor	输出一个用于选择标注颜色的选择框。	fontOptions	<mapxtreme:fontColor />

元素	说明	父级	**JSP** 语法
fontHaloColor	输出一个用于选择晕环颜色的选择框。	fontOptions	<mapxtreme:fontHaloColor />
fontBold	输出一个复选框，用于更改标注是否为粗体。	fontOptions	<mapxtreme:fontBold />
fontItalic	输出一个复选框，用于更改标注是否为斜体。	fontOptions	<mapxtreme:fontItalic />
fontUnderline	输出一个复选框，用于更改标注是否为下划线。	fontOptions	<mapxtreme:fontUnderline />
专题	用于专题向导的根标记。定义此标记是否在对话框中。如果提供图层，那么系统将不提示用户选择图层。如果提供了图层和列，则用户将只能更改范围专题选项。有关示例请参阅 thematicdialog.jsp。	mapapp	<mapxtreme:thematic [dialog=”true|false”] [layer=”layerIndex”] [column=”columnIndex”]>…</ mapxtreme:thematic>
themelayer	如果没有选择图层，将对此标记的内容求值。	专题	<mapxtreme:themelayer />
themecolumn	如果没有选择列，将对此标记的内容求值。	专题	<mapxtreme:themecolumn />
themeoptions	如果已经选择图层和列，且列为数字，那么将对此标记的内容求值。	专题	<mapxtreme:themeoptions />
themeStartColor	输出一个用于选择起始颜色的选择框。	themeoptions	<mapxtreme:themeStartColor [selected=”color”] />
themeEndColor	输出一个用于选择结束颜色的选择框。	themeoptions	<mapxtreme:themeEndColor [selected=”color”] />
themeDistributionType	输出一个选择框，用于选择范围专题的分布方法。	themeoptions	<mapxtreme:themeDistributionType />
themeBreaks	输出一个文本框，用于选择有多少个分点值用于范围专题。	themeoptions	<mapxtreme:themeBreaks />
themeLayerList	输出一个选择框，用于选择执行专题所在的图层。	themelayer	<mapxtreme:themeLayerList />

元素	说明	父级	**JSP** 语法
themeColumnList	输出一个选择框，用于选择执行专题所在的列。	themecolumn	<mapxtreme:themeColumnList />
themetool	输出一个用于打开专题对话框的工具栏工具。	toolbar	<mapxtreme:themetool [page=”relativeURL”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] [options=”JavaScript Options” />
legendtool	输出一个用于打开图例对话框的工具栏工具。	toolbar	<mapxtreme:legendtool [page=”relativeURL”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] [options=”JavaScript Options” />
legendlist	可见于各个图层、专题和输出图例的 legendelement 的设置脚本变量。有关示例请参阅 legenddialog.jsp。	mapapp	<mapxtreme:legendlist [showRangedThemes=”true|false”] [showIndividualValueThemes=”true|false”] [showOverrideThemes=”true|false”] [showSelectionThemes=”true|false”]>…</mapxtreme:legendlist>
legendelement	输出图例。	legendlist	<mapxtreme:legendelement [titleFont=”font”] [titleFontSize=”size”] [font=”font”] [fontSize=”size”] />
legend	输出给定图层和专题的图例。可指定图层和专题的索引或名称。	mapapp	<mapxtreme:legend layer=”layerIndex” theme=”themeIndex” layerName=”layerName” themeName=”themeName” [title=”title”] [titleFont=”font”] [titleFontSize=”size”] [font=”font”] [fontSize=”size”] />
printpreviewtool	输出一个工具栏工具，用于输出预览对话框。	toolbar	<mapxtreme:printpreviewtool [page=”relativeURL”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] [options=”JavaScript Options” />

元素	说明	父级	**JSP** 语法
tableinfo	可见于给定图层的所有记录。如果未指定图层，则将在入局请求中查找该标记。行数指定每次显示的记录数量。	mapapp	<mapxtreme:tableinfo [layer=”layerIndex”] [rows=”numOfRows”]>…</ mapxtreme:tableinfo>
rowNum	输出当前行号。	tableinfo	<mapxtreme:rowNum />
tableColumns	可见于给定图层的列。如果忽略图层，则将在 tableinfo 标记中查找该图层。	mapapp 或 tableinfo	<mapxtreme:tableColumns [layer=”layerIndex”] >…</ mapxtreme:tableColumns>
columnValue	输出当前记录和列的值。	tableColumns 和 tableinfo	<mapxtreme:columnValue [layer=”layerIndex”] [column=”columnIndex”] />
columnName	输出当前列的名称。	tableColumns	<mapxtreme:columnName [layer=”layerIndex”] [column=”columnIndex”] />
nextTableInfo	输出按钮，用于转至下一组记录。	mapapp	<mapxtreme:nextTableInfo />
zoomNumeric	地图显示标记的根标记。定义此标记是否在对话框中。有关示例请参阅 mapdisplaydialog.jsp。	mapapp	<mapxtreme:zoomNumeric [dialog=”true|false”]>…</ mapxtreme:zoomNumeric>
zoomRange	输出地图的当前缩放范围。如果 readOnly 设置为 false，则将输出文本框。设置四舍五入为 10 的因数，确定如何对值进行舍入。忽略表示不对数字进行舍入。	zoomNumeric	<mapxtreme:zoomRange [size=”size”] [readOnly=”true|false] [round=”num”] />
distanceUnits	输出用于当前地图距离的单位。	zoomNumeric	<mapxtreme:distanceUnits />
centerX	输出地图中心的 x 坐标。如果 readOnly 设置为 false，则将输出文本框。设置四舍五入为 10 的因数，确定如何对值进行舍入。忽略表示不对数字进行舍入。	zoomNumeric	<mapxtreme:centerX [size=”size”] [readOnly=”true|false] [round=”num”] />

元素	说明	父级	JSP 语法
centerY	输出地图中心的 y 坐标。如果 readOnly 设置为 false，则将输出文本框。设置四舍五入为 10 的因数，确定如何对值进行舍入。忽略表示不对数字进行舍入。	zoomNumeric	<mapxtreme:centerY [size=”size”] [readOnly=”true|false] [round=”num”] />
zoomnumerictool	输出一个工具栏工具，用于打开缩放数值对话框。	toolbar	<mapxtreme:zoomnumerictool [page=”relativeURL”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] [options=”JavaScript Options” />
viewEnirelayer	用于查看整个图层标记的根标记。定义此标记是否在对话框中。有关示例请参阅 viewentirelayerdialog.jsp。	mapapp	<mapxtreme:viewEntireLayer [dialog=”true|false”>…</ mapxtreme:viewEntireLayer>
viewEntireLayerList	输出一个选择框，用于选择要缩放到的图层。	viewEntireLayer	<mapxtreme: viewEntireLayerList />
viewentirelayertool	输出一个工具栏工具，用于打开查看整个图层的对话框。	toolbar	<mapxtreme:viewentirelayertool [page=”relativeURL”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] [options=”JavaScript Options” />
overviewmap	输出当前地图的概览。	mapapp	<mapxtreme:overviewMap width=”size” height=”size” [dialog=”true|false”] [zoomOutFactor=”factor”] [zoomOut=”zoomOutAmount”] [showPoints=”true|false”] [showPolylines=”true|false”] [showPolygons=”true|false”] [mapFile=”file”] [mapPath=”path”] [backgroundColor=”FFFFFF”] [numOfColors=”256”] />
projection	投影标记的根标记定义此标记是否在对话框中。有关示例请参阅 projectiondialog.jsp。	mapapp	<mapxtreme:projection [dialog=”true|false”]>…</ mapxtreme:projection>
projectionCategories	输出一个选择框，列出所有投影种类。	projection	<mapxtreme: projectionCategories [size=”1”] />

元素	说明	父级	**JSP** 语法
projectionMembers	输出一个选择框，列出当前种类的所有投影。	projection	<mapxtreme:projectionMembers [size=”5”] />
changeProjection	更改所选投影的投影。	projection	<mapxtreme:changeProjection [value=”text”] [src=”relativeURL”] />
projectiontool	输出一个工具栏工具，用于打开投影对话框。	toolbar	<mapxtreme:projectiontool [page=”relativeURL”] [alt=”text”] [img=”relativeURL”] [width=”size”] [height=”size”] [options=”JavaScript Options” />
featureSet	可见于给定 FeatureSet 的所有图元。	mapapp	<mapxtreme:featureSet featureSet=”FeatureSet”>…</mapxtreme:featureSet>
feature	可见于给定图元的所有属性。	featureSet or mapapp	<mapxtreme:feature [feature=”Feature”] [featureSet=”FeatureSet”] >…<mapxtreme:feature>
featureName	输出当前属性的名称。	feature	<mapxtreme:featureName />
featureValue	输出当前属性的值。	feature	<mapxtreme:featureValue />
svgmap	渲染 SVG 文档。（SVGMapBean 类必须位于 Requesthandler 的列表之内。）	mapapp	<misvg:svgmap [width=”size”] [height=”size”] />
visible	如果已经加载图层但是图层为关闭，则无需发出服务器请求。	layername	<misvg:visible />
highlight	调用 onmouseover 事件，更改颜色并向该项目添加下落阴影。	layername	<misvg:highlight />

创建定制 JSP 标记

MapXtreme Java 利用 Java 服务器页技术快速开发和部署 Web 应用程序。我们在此提供了定制 JSP 标记库，使用文本编辑器或 IDE 即可将其插入到 .JSP 文件中。

这些标记在 MapXtreme Java 管理器的 Web 应用程序构建器中显示为窗口部件，在其中可以选择所需元素，并将其添加到要另存为 .JSP 文件的布局框中。在运行时， .JSP 将与执行应用程序业务逻辑的 servlet 通信。如果有必要更改应用程序的显示，可在 Web 应用程序构建器中轻松重排、添加或删除窗口部件，创建新的 .JSP，而不会影响到 servlet 的内容生成操作。

这些定制标记设计用于 MVC （模型 / 视图 / 控制器） JSP-servlet 体系结构。所生成的 .JSP （视图）包含表格，相应表格提交到一般 servlet （控制器）。控制器 servlet 重定向至执行类似创建专题、执行半径搜索等必要业务逻辑的适当 Java bean （模型），并将相应请求转发回到 .JSP 文件，由后者显示更新的地图。

视图

取决于所需的标记功能，在此提供了若干基类。所有这些类的根类均为 MITag。这一抽象类实施 JSP 规范中定义的 javax.servlets.jsp.tagext.Tag 接口。此外还提供了实用程序方法，用于获取 ServerProperties 和 MappingSession 对象并执行某些公共任务。

模型

模型类基于用户从定制标记的输入来执行预期的业务逻辑。每个模型类的侧重都应该非常明确。例如， MapXtreme 提供了一个 bean 类用于放大 (com.mapinfo.jsptags.mapxtreme.ZoomInToolBean)，提供了另一个类用于缩小 (com.mapinfo.jsptags.mapxtreme.ZoomOutToolBean)。尽管存在将两者合并到一个缩放 bean 中的可能，但我们还是选择了分别提供，以便实现更加优良的模块化设计。

要创建模型类，只需实施 com.mapinfo.jsptags.TagBean 接口即可。实施方法有三种。第一， getParameterKey 应该只返回表示该类的唯一字符串。这将允许 RequestHandler servlet 正确确定用于给定请求的 TagBean 类。第二是 setServerProperties。此方法允许用户获取有关给定应用程序的各种服务器设置的引用。尽管此方法可以多次调用 （如服务器设置在应用程序运行期间更改），但是务必确保在任何客户机请求之前进行调用。最后是实施进程。此方法应该执行必要的业务逻辑。提供 MappingSession、 HttpServletRequest 和 HttpServletResponse 作为调用参数。该方法的返回值将表示在处理方法返回之后，如何处理 RequestHandler servlet。返回 true 将导致 RequestHandler servlet 执行任意清空和返回。返回值为 false 导致 RequestHandler servlet 将用户重定向到 JSP 或 HTML 页。

在完成之后，必须将 TagBean 类注册到 RequestHandler servlet。为此只需将完全合格的类名添加到 Beans 初始化参数即可。

控制器

作为定义标记的作者，无需编写任何 servlet。 RequestHandler servlet 将接受入局请求，创建适当的 MappingSession 对象，将控制传递给 TagBean 类用于处理，且可选将客户机重定向到新视图。为执行这些任务， RequestHandler servlet 将查找特定的 HTML 参数。这些参数通常以隐含表单字段传递，并可被添加到 URL。下表对这些参数作出了说明：

名称	必需	说明
appname	是	这是用于应用程序的唯一名称。可作为 mapapp 标记属性覆盖。 **Note:** 该名称存储为名为 APP_NAME_PARAMETER 的 MappingSession 类的常量。
mapinfobean	是	应该处理入局请求的 TagBean 类的名称。该值应该匹配预期类的 getParameterKey 方法返回的值。 **Note:** 该名称存储为名为 PARAMETER_KEY_NAME 的 TagBean 类的常量。
redirect	否	设置覆盖任意默认值的返回 JSP 或 HTML 页。如果没有指定重定向，用户将重定向到 MappingSession.getDisplayURL()。
debug	否	输出有关 RequestHandler 的调试信息。只提供用于调试目的，不应在应用程序中使用。

此外，提供了实用程序方法来帮助用户构建必要的语法，在视图和模型之间传递参数值。对于 TagBeans 的自助重定向控制 （从进程返回 true），可使用 RequestHandler servlet 上的 getEncodedURL 方法之一。扩展 MITag 类之一的定制标记具有若干种获取必需参数的方法。对于 HTML 表单，可使用 getRequiredFieldsAsHTML 方法。此方法将返回包含隐含 HTML 表单字段的字符串，该字段包含了必要的参数。对于要处理源自 URL （如 href 标记）的情况，可使用 getRequiredFieldsAsQueryString 方法。

创建添加 TAB 图层标记

本节说明使用定制标记扩展 JSP 库的必要步骤。在示例中，将创建添加 TAB 图层标记。这将允许用户向 MapJ 添加 TAB 图层。本示例的源代码随 MapXtreme Java 的安装提供。

对于此特性，我们希望为用户提供 tab 文件列表和新图层的位置，在该列表中可选择要添加到 MapJ 的文件。 JSP 作者将确定相应的目录。这需要三个标记。

视图

首先，创建一个生成 HTML 组合框的标记。此类将扩展 MITag 并实施 doStartTag()。此方法将获取给定目录中的文件列表，并输出 <option> 标记，用于每个具有 .tab 文件扩展名的文件。此外，还将实施名为 setDirectory(String dir) 的方法。这将允许我们定义用于该标记的属性，允许 JSP 的作者确定所要使用的目录。

接下来，我们需要一个标记来允许用户输入将图层添加到 MapJ 的位置。这一操作将只创建一个 HTML 文本框。我们还将实施 setPosition 和 setSize，以允许 JSP 作者将文本框的默认位置和大小定义为属性。

最后，我们将创建一个标记，生成 HTML 表单标记以及必需的参数。前两个标记将包含在此标记之中。所生成的 JSP 将如下所示：

```
<addtablayer:addTabLayer>
<p>File: <addtablayer:tabFileChooser directory="c:/maps" />
<br>Position: <addtablayer:tabPosition size="5" />
<br><input type=submit value="Add Layer" >
</addtablayer:addTabLayer>
```

为在 JSP 页中使用这些标记，我们需要创建标记库定义 (TLD)。 addtablayer_4_7.tld 中提供了用于这些标记的 TLD。

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
"http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
<taglib>
    <tlibversion>1.0</tlibversion>
    <jspversion>1.1</jspversion>
    <shortname>addtablayer</shortname>
    <uri></uri>
<info>Custom tags to add a tab layer.</info>
<tag>
<name>addTabLayer</name>
<tagclass>
com.mapinfo.jsptags.mapxtreme.AddTabLayerTag
</tagclass>
```

```
            <bodycontent>JSP</bodycontent>
        </tag>
        <tag>
            <name>tabFileChooser</name>
        <tagclass>
    com.mapinfo.jsptags.mapxtreme.AddTabLayerFileChooserTag
    </tagclass>
            <bodycontent>JSP</bodycontent>
            <attribute>
                <name>directory</name>
                <required>true</required>
                <rtexprvalue>false</rtexprvalue>
            </attribute>
        </tag>
        <tag>
            <name>tabPosition</name>
        <tagclass>
    com.mapinfo.jsptags.mapxtreme.AddTabLayerPositionTag
    </tagclass>
            <bodycontent>JSP</bodycontent>
            <attribute>
                <name>size</name>
                <required>false</required>
                <rtexprvalue>false</rtexprvalue>
            </attribute>
            <attribute>
                <name>position</name>
                <required>false</required>
                <rtexprvalue>false</rtexprvalue>
            </attribute>
        </tag>
    </taglib>
```

Within the JSP page, add the following taglib directives:

```
<%@ taglib uri="/addtablayer" prefix="addtablayer" %>
<%@ taglib uri="/zoomtofeature" prefix="zoomtofeature" %>
```

模型

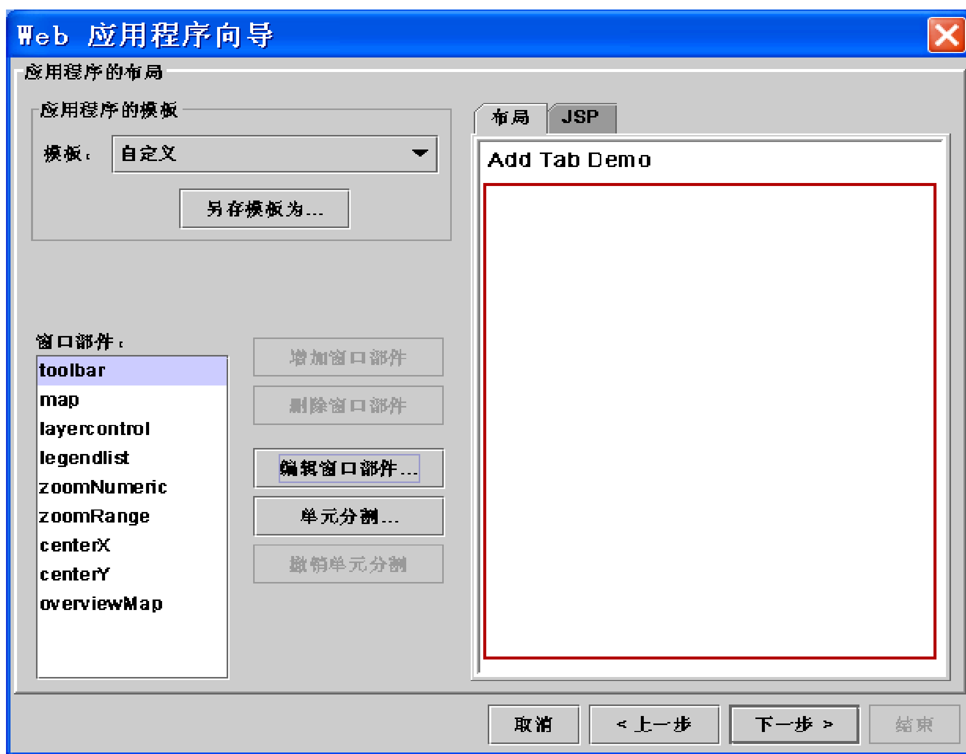
要处理用户的请求，我们需要获取文件名和位置，创建 DataProviderHelper、TableDescHelper 和 DataProviderRef，以将其添加到 MapJ。我们需要从 MappingSession 获取对于应用程序的 MapJ 的引用。源代码随 MapXtreme Java 的安装提供。

控制器

控制器所需的唯一步骤是将 TagBean 类注册到 RequestHandler。只需将完全合格的类名 (com.mapinfo.jsptags.mapxtreme.AddTabLayerBean) 添加到 Beans 初始化参数列表即可。

向 Web 应用程序向导添加定制标记

在完成定制标记之后，可能需要将标记集成到 Web 应用程序向导中。通过这一操作，任何使用 Web 应用程序向导来构建 JSP 应用程序的用户均可以通过点击来使用相应定制标记。



要将定制标记集成到 Web 应用程序向导中，请执行以下操作：

1. 创建一个包含定制标记的所有 .class 文件的 .jar 文件。
   可使用随 Java SDK 提供的 jar 实用程序来完成此操作。

类似 WinZip 的第三方软件实用程序也可用于构建 jar 文件；但使用时务必仔细设置适当的选项，以便在 .jar 文件中保留类文件的文件包 （路径）。（如 WinZip 用户在向存档文件添加 .class 文件时，可能需要选择 "Save Extra Folder Info" 复选框。）

2. 在安装 servlet 容器的过程中，在以下目录中创建一个新的目录：webapps/mapxtreme47/client。例如，运行 Tomcat 的 Windows 用户可能需要创建以下目录，用于部署添加 TAB 标记：

```
D:\mapxtreme\tomcat\webapps\mapxtreme47\client\
addtab
```

3. 将 .jar 文件和 addtablayer_4_0.tld 文件复制到刚刚创建的目录中。
4. 将 .jar 文件复制到 mapxtreme47 上下文的 lib 目录中：

```
webapps/mapxtreme47/WEB-INF/lib
```

5. 关闭 servlet 容器。
6. 编辑 webapps/mapxtreme47/WEB-INF/manager.xml 文件，添加 taglib 块。对于要显示在 Web 应用程序向导中的每个定制标记，在此 taglib 块中均有一个对应的标记块。

   在此可能需要制作 manager.xml 的备份，以便在编辑出错时使用。

以下摘录代码显示了 manager.xml 文件中的相应部分。本示例假定您在第 2 步中创建的目录名为 addtab：

```
<panel class="com.mapinfo.mjm.client.mapdefs.MapDefPanel"
    required-services="1" />
  <panel class="com.mapinfo.mjm.client.appbuilder.AppBuilderPanel" />
  <panel class="com.mapinfo.mjm.client.namedres.NamedResourcesPanel"
    required-services="9" />
        <!-- 3rd-party JSP custom tag libraries -->
  <taglib uri="/client/addtab/addtablayer_4_0.tld">
    <tag name="addTabLayer" required_services="1"
        allowed_context="1" />
    <tag name="tabFileChooser" required_services="1"
        allowed_context="1" />
    <tag name="tabPosition" required_services="1"
        allowed_context="1" />
  </taglib>
    <!-- Mapping preferences specified in the MJM Client -->
  <preferences maps_dir="D:/data/maps" mime_type="image/gif"
    is_remote="true" />
  <!-- Recent maps loaded in MJM Client -->
  <recent-maps>
```

添加到 manager.xml 的文本包含对于标记名称 （在 .tld 文件中定义）的引用以及对于 .tld 文件名的引用。如果重命名 .tld 文件，或者如果重命名标记，则需要编辑 manager.xml 匹配相应的名称。

7. 保存对 manager.xml 所做编辑，然后重新启动 servlet 容器。
8. 运行 MapXtreme Java 管理器，然后启动 Web 应用程序向导。向 manager.xml 添加的每个标记均将显示在可用标记的列表中。

定制标记属性

对于 <taglib> 条目，需要指定以下属性：

属性	必需	说明
uri	是	到 .tld 文件的路径，该路径是相对于 mapxtreme47 上下文的根的路径。

对于每个 <tag> 条目，需要指定以下属性：

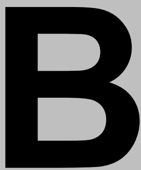
属性	必需	说明
name	是	定制标记的名称 （和显示在 .tld 文件中的名称相同）。
required_services	是	说明此标记所需的可用服务 （在类 com.mapinfo.mjm.MJMServiceType 中定义）。值为 1 表示此标记正常工作需要 MapXtreme Java 服务。如果需要多个服务，则此值将表示所需各个服务值的总和。
allowed_context	是	说明允许此标记的应用程序上下文 － com.mapinfo.jsptags.TagHandler 中的常数说明了可能用于此设置的值。值为 1 表示在应用程序的页面级别允许使用该标记 （如该标记将出现在可能添加到应用程序构建器向导中应用程序布局的窗口部件列表中）。
icon	否	到图标 （通常为 .gif）文件的路径，存储在一个 .jar 文件中。该路径相对于其所在 .jar 文件的根。可用于说明应用程序构建器向导中的标记，并且在标记表示要在工具栏中使用的标记时尤为实用。
editor	否	用于标记的定制 GUI 编辑器类的类名。该名称只应指定用于更加复杂的标记，在所提供的默认 GUI 编辑器无法满足要求的情况下使用。参阅下一节。

添加用于定制标记的定制 GUI 编辑器

在 “编辑” Web 应用程序构建器向导中的定制编辑窗口部件的属性时，通常提供给您的只是用于由定制标记所示属性的不同的值。一般由 Web 应用程序构建器向导提供的默认 GUI 编辑器应该可以满足编辑这些属性的要求。但是，如果标记比较复杂，且默认的编辑器不能展示所需全部信息 （或展示过多信息），那么则需要用户自行提供相应的编辑器。

第一步是创建您自己的 com.mapinfo.mjm.util.CustomTagEditor 类 （是 javax.swing.JPanel 的扩展）的实施。这一面板将包含编辑定制标记属性所必需的 GUI 控件。有关为使之正常工作而必须实施的方法的详细信息，请参阅 CustomTagEditor 类的 Javadoc。

在具备 CustomTagEditor 类之后，只需在 manager.xml （参阅以上说明）中指定该类作为 <tag> 条目的编辑器属性即可。



理解 MapBasic 样式字符串

MapBasic 字符串是 MapXtreme Java 可以读取每个图元样式或每个图元标签样式的途径之一。（XML 文档是另一种方式）。本附录介绍如何创建用于以下 5 种样式类型的 MapBasic 样式字符串：画笔、画刷、符号、字体和颜色。本附录的小结中说明了对于使用 MapInfo Professional 的 MapBasic 窗口的样式，如何确定所用的 MapBasic 字符串。

本附录内容：

画笔样式

画笔样式也称为直线样式，在 MapBasic 中用于指定直线、弧线或区域边界线性等对象的宽度、图案和颜色。 MapBasic 画笔子句采用以下语法：

```
PEN ( 宽度 , 图案 , 颜色 )
```

宽度可以通过屏幕像素或磅值来表示。以介于 1 至 7 之间的数值表示宽度的屏幕像素。

介于 11 和 2,047 之间的数值减 10 之后，以磅的十分之一 (1/10 pt) 来表示。 MapBasic 提供了用于转换的函数：PointsToPenWidth() 和 PenWidthToPoints()。

0 仅限于画笔图案为用于不可见直线的 1 时有效。

图案 是从 1 至 118 的整数，图案编号 1 表示不可见。图案编号和画笔文件中的画笔编号相对应。后续内容中提供了画笔样式的图解。

注： 目前， MapXtreme Java 中不支持特定的画笔样式。其中包括在段的一端或两端具有标记的画笔。尤其是 59、60、61、62、78、79、80、86、87、88、94、95、96、102、 103 和 104 号画笔。

颜色是一个表示 24 位 RGB 颜色值的整数。有关颜色的详细信息，请参阅第 379 页的*颜色* 。

表示黑色铁轨的 MapBasic 字符串， 2 个像素宽，如下所示：

```
Pen (2, 118, 0)
```

下表列出了可用的默认直线样式：

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60

61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90

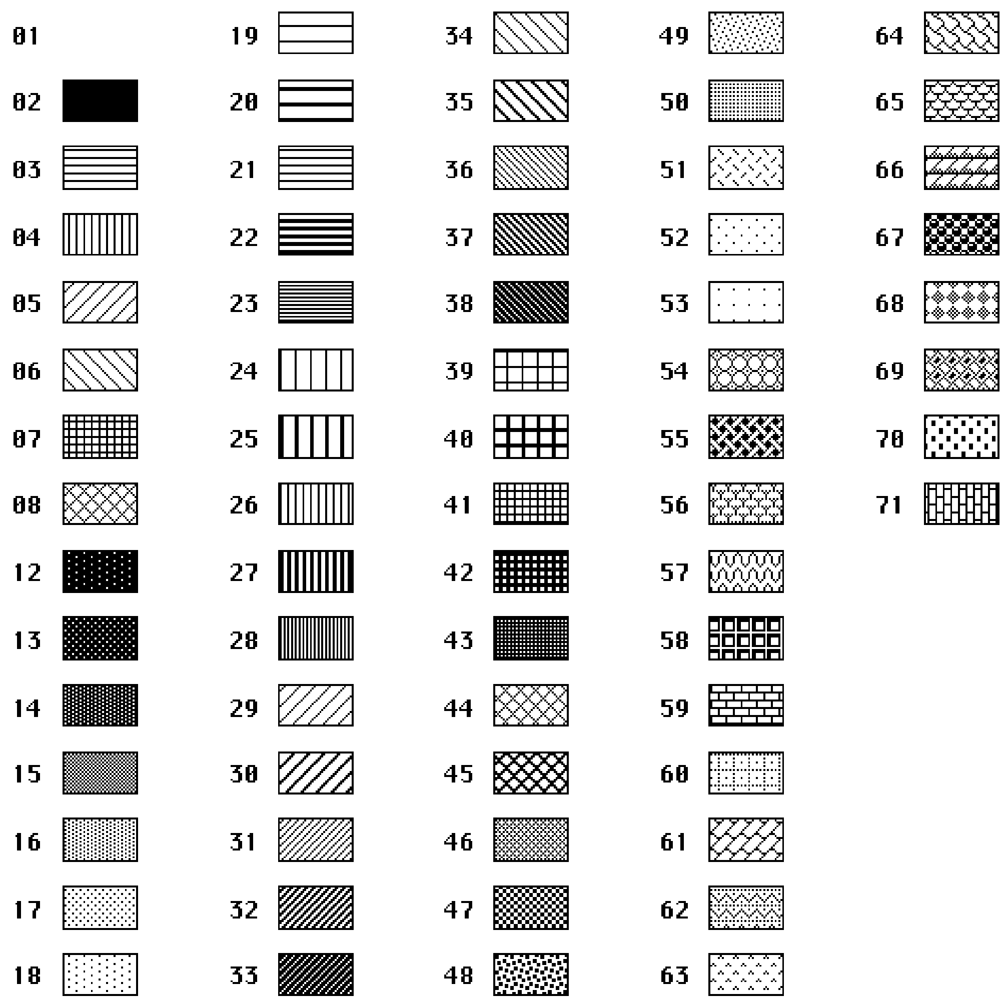
91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118

画刷样式

画刷样式指定了圆或区域等填充对象的图案、前景颜色和背景颜色。画刷子句采用以下语法：

```
Brush ( 图案 , 前景色 [, 背景色 ])
```

图案是一个从 1 至 71 的编号。图案编号 1 为 “不填充”，图案编号 2 为实体填充。图案编号 9-11 为保留。下表介绍了可用的样式。



前景色和背景色参数均为整数，表示 24 位 RGB 的颜色值。有关颜色的详细信息，请参阅第 379 页的*颜色* 。

要指定透明填充样式，可使用图案编号 3 或更大编号，同时忽略背景色参数。例如：
Brush(5, 255)

符号样式

符号子句指定点对象的外观。以下有三种不同形式的符号子句：

- *符号子句 － MapInfo 3.0 兼容语法*
- *符号子句 － TrueType 字体语法*
- *符号子句 － 定制图像文件语法*

符号子句 － MapInfo 3.0 兼容语法

要使用 “MapInfo 的旧符号”（在此前版本的 MapInfo 中使用的符号）指定符号样式，可使用以下语法：

```
SYMBOL ( 形状 , 颜色 , 大小 )
```

形状参数是一个整数值，大小为 31 或更大， 31 表示空白符号 （如对象不可见）。符号的标准集包括符号 32 至 67 （含上、下限）。通过使用 MapInfo Professional 中的符号标记工具，可从符号集中添加、编辑或删除符号。

颜色参数是一个表示 24 位 RGB 颜色值的整数。有关颜色的详细信息，请参阅第 379 页的*颜色* 。

大小参数一个从 1 至 48 的整数，表示磅值。

下表列出了随 MapInfo 3.0 兼容符号集提供的默认符号。

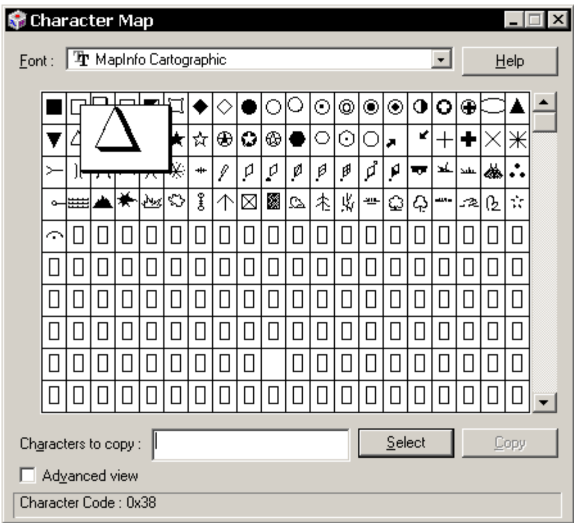
31	41	51	61
32	42	52	62
33	43	53	63
34	44	54	64
35	45	55	65
36	46	56	66
37	47	57	67
38	48	58	
39	49	59	
40	50	60	

符号子句 — TrueType 字体语法

要基于 TrueType 字体中的字符指定符号样式，可使用如下语法：

```
SYMBOL （形状， 颜色， 大小， 字体名， 字体样式， 旋转）
```

形状 是字体集中的字符的 ASCII 值。要确定 TrueType 符号的形状值，可在操作系统的字体查看器工具中查看字体集，例如用于 Windows 的 Unicode 字符映射的字体查看工具。注意用于所选符号的字符代码。



颜色 参数是一个表示 24 位 RGB 颜色值的整数。有关颜色的详细信息，请参阅第 379 页的*颜色*。

大小参数一个从 1 至 48 的整数，表示磅值。

字体名称参数是一个文本字符串，用于确定字体名称 （如 Wingdings）。

字体样式参数是一个控制设置的整数，例如粗体。下表列出了可以用作字体样式的值：

字体样式值	符号样式效果
0	纯文本
1	粗体文本
16	符号采用黑色边框
32	下落阴影
256	符号采用白色边框

要指定两种或多种样式属性，可从左侧列中添加相应的值。例如，要指定 “粗体” 和 “下落阴影”，可使用 33。

旋转参数是浮点数，表示旋转的角度数。

MapXtreme Java TrueType 符号

地图箭头字体

示例 ID："mistyles/fonts/Map Arrows/33"

	33 - 40
	41 - 42

地图制图字体

示例 ID："mistyles/fonts/Map Cartographic/33"

	33 - 40
	41 - 50
	51 - 60
	61 - 70
	71 - 80
	81 - 90
	91 - 100
	101 - 110
	11-113

其他地图字体

示例 ID："mistyles/fonts/Map Miscellaneous/33"

	33 - 40
	41 - 50
	51 - 60
	61 - 70
	71 - 80
	81 - 90
	91 - 100
	101 - 110
	111 - 120
	121 - 130
	131 - 140

油气专用地图字体

示例 ID："mistyles/fonts/Map Oil&Gas/33"

	33 - 40
	41 - 50
	51

房地产专用地图字体

示例 ID："mistyles/fonts/Map Real Estate/33"

	33 - 40
	41 - 50
	51 - 60
	61 - 70
	71 - 80
	81 - 85

地图符号

示例 ID："mistyles/fonts/Map Symbols/33"

■ ◆ ● ★ ▲ ▼ □ ◇	33 - 40
	41 - 50
	51 - 60
	61 - 68

MapInfo 符号

示例 ID："mistyles/fonts/MapInfo Symbols/33"

■ ◆ ● ★ ▲ ▼ □ ◇	33 - 40
	41 - 50
	51 - 60
	61 - 68

交通专用地图字体

示例 ID："mistyles/fonts/Map Transportation/33"

	33 - 40
	41 - 50
	51 - 60
	61 - 70
	71 - 80
	81 - 90
	91 - 100
	101 - 110
	111 - 119

天气专用地图字体

示例 ID："mistyles/fonts/Map Weather/33"

	33 - 40
	41 - 50
	51 - 60
	61 - 64

符号子句 — 定制图像文件语法

要基于定制图像文件指定符号样式，例如 .gif 或 .bmp 文件，可使用如下语法：

```
SYMBOL ( 文件名 , 颜色 , 大小 , 定制样式 )
```

文件名参数是一个文本字符串，确定 CustSymb 目录中的位图文件 （如 Arrow.BMP）。

颜色参数是一个表示 24 位 RGB 颜色值的整数。有关颜色的详细信息，请参阅第 379 页的*颜色* 。

MapXtreme Java 中不支持大小参数。定制图像将以其实际的大小显示。

定制样式参数是一个整数，控制是否使用颜色和背景属性。下表列出了可以用作定制样式的值：

定制样式值	符号样式效果
0	“显示背景”设置和 “应用颜色”设置均为禁用，符号以其默认状态显示。位图中的白色像素显示为透明，允许显示符号之后的对象。
1	“显示背景”设置为启用，图像中的白色像素为不透明。
2	“应用颜色”设置为启用，图像中不是白色的颜色将为符号颜色值所取代。
3	“显示背景”和 “应用颜色”设置均为启用。

有关随 MapXtreme Java 提供的定制 .gif 符号小结，请参阅*附录 D: 系统属性* 。

字体样式

MapBasic 字体子句指定了 MapXtreme Java 中标记的外观 （字型、颜色等）。字体子句采用以下语法：

```
FONT ("字体名", 样式, 大小, 前景色 [, 背景色] )
```

双引号之间的字体名是要显示的字型。

样式是应用于下表所示字型的属性。

大小在 MapBasic 字符串中必须为 0，由于地图上的每个标记均附属到地图本身 （因此文本大小可以随之缩放）。

前景色是一个表示 24 位 RGB 颜色值的整数。有关颜色的详细信息，请参阅第 379 页的*颜色* 。

背景色为可选，如果包括此参数，则标记之后的区域将使用您所指定的颜色填充。有关颜色的详细信息，请参阅第 379 页的*颜色* 。

样式值	字体外观效果
0	纯文本
1	粗体
2	斜体
4	下划线
16	轮廓线 （仅限 Macintosh 支持）
32	阴影
256	晕环
512	全部大写
1024	展开

要指定两种或多种样式属性，从左侧列中添加相应的值。例如，要指定 “粗体”和 “全部大写”，可使用 513。

颜色

颜色通常以红、绿、蓝的相对浓度来定义。每种颜色均为从 0 至 255 的数字 （含上、下限），颜色的 RGB 值均通过以下公式计算：

(红色 * 65536) + (绿色 * 256) + 蓝色

常用的颜色以及值如下所示：

红：16711680

浅绿：65280

深绿：32768

蓝：255

青：65535

洋红：16711935

黄：16776960

黑：0

白：16777215

深灰：8421504

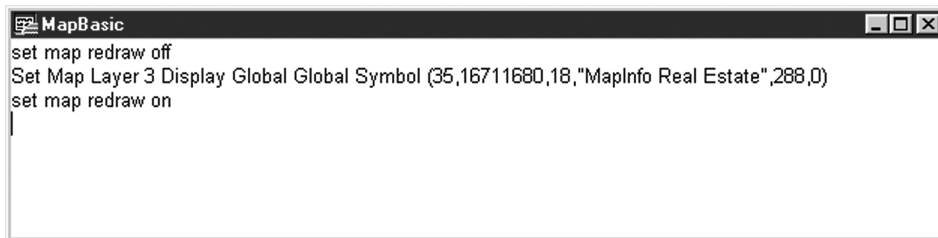
浅灰：12632256

使用 MapInfo Professional 确定 MapBasic 样式

只要可以访问 MapInfo Professional，就能为 MapXtreme Java 的每图元样式和每图元标注样式中可用的任意画笔、画刷或符号样式确定 MapBasic 字符串语法。

在显示 MapBasic 窗口的情况下使用 MapInfo Professional。从图层控制，设置用于区域、线或点对象的覆盖样式。在 MapBasic 窗口中查看样式的 MapBasic 语法。

例如，下图显示了使用 MapInfo 房地产专用 TrueType 字体集，以带有白色光晕、下落阴影和无旋转效果的 18 号红色字体显示房屋符号的必要语法。



C

MAPINFO.MAPINFO_ MAPCATALOG

本附录介绍 MAPINFO.MAPINFO_MAPCATALOG。

本附录内容：

概览

MAPINFO.MAPINFO_MAPCATALOG 是一个数据库表，用于存储数据库中有关几何表的元数据。 MAPINFO.MAPINFO_MAPCATALOG 使用表名和所有者名字作为关键字，标识几何列、几何类型、坐标系、最小边界矩形以及表和图元级别的样式信息。

除了 MapXtreme Java 之外，还有访问数据库地图数据的其他 MapInfo 产品也使用 MAPINFO.MAPINFO_MAPCATALOG。

如果使用 EasyLoader 实用程序将 MapInfo TAB 文件上载到数据库，则 MapCatalog 将自动根据该表更新。此前的附录中提供了 EasyLoader 的说明。如果表通过其他方式创建，则 MapCatalog 可通过手动或通过定制应用程序更新。

尽管众多 MapInfo 产品都需要使用 MAPINFO.MAPINFO_MAPCATALOG，但是如果表定义图层的 TableDescHelper 构造器提供了需要从 MAPINFO.MAPINFO_MAPCATALOG 获取的数据，则 MapXtreme 就无需再使用该表。尤其需要注意的是，该表需要几何列名、坐标系和样式列信息。缺少上述任意数据都将令 MapXtreme 查询 MAPINFO.MAPINFO_MAPCATALOG，如果数据不可用则将会报错。

MapXtreme Java SQL 脚本

在数据库没有 MAPCATALOG，并且用户无法访问 MapInfo Professional 的情况下，我们专门为此提供了创建 MAPCATALOG 的 SQL 脚本。在安装之后，以下文件位于 /MapXtreme-4.7.0/examples/server/scripts 目录之下：

- oracle_mapcat.sql － 用于 Oracle
- informix_mapcat.sql － 用于 Informix Universal Server

这些脚本创建了一个名为 MAPINFO 的用户，随后创建了一个包含以下三个字符串列的 MAPCATALOG：RENDITIONCOLUMN、 RENDITIONTYPE 和 RENDITIONTABLE。这些列支持 MapXtreme Java 在每个图元的基础上存储样式信息。

RENDITIONCOLUMN 包含表中的列名，该表保存了图元的样式信息；如果没有样式列，则为空。

RENDITIONTYPE 列包含一个枚举值，表示 RENDITIONCOLUMN 如何由 MapXtreme Java 解释。支持的类型如下：

- 没有每个图元列出现：0
- MapBasic 样式：1
- MapXtreme Java XML 样式：2

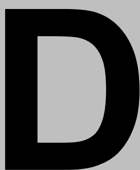
RENDITIONTABLE 列保留为日后使用。

MAPINFO.MAPINFO_MAPCATALOG 格式

MAPINFO.MAPINFO_MAPCATALOG 表的结构如下所示：

SPATIALTYPE	FLOAT
TABLENAME	CHAR(32)
OWNERNAME	CHAR(32)
SPATIALCOLUMN	CHAR(32)
DB_X_LL	FLOAT
DB_Y_LL	FLOAT
DB_X_UR	FLOAT
DB_Y_UR	FLOAT
COORDINATESYSTEM	CHAR(254)
SYMBOL	CHAR(254)
XCOLUMNNAME	CHAR(32)
YCOLUMNNAME	CHAR(32)
RENDITIONTYPE	INTEGER
RENDITIONCOLUMN	CHAR(32)
RENDITIONTABLE	CHAR(32)
NUMBER_ROWS	INTEGER

系统属性



本附录列示 MapXtreme Java 支持的系统属性。

属性

下表列出了和各个主题关联的属性。

JDBC 连接池

键	com.mapinfo.connpool
说明	指定提供定制 JDBC 连接池的 Java 类。
值	实施 com.mapinfo.dp.conn.ConnectionPool 接口的类的完全类规范。
默认值	如未指定，默认连接池将由 MapXtreme Java 提供。

字体渲染

键	com.mapinfo.renderer.quality
说明	指定是否使用较高质量（但消耗更多系统资源）的方法渲染字体。
值	值为 true 表示高质量的字体，值为 false 表示普通质量的字体
默认值	false

数据绑定

键	com.mapinfo.max.where.items
说明	指定在单一数据绑定 SQL 查询中可以存在的 WHERE 条件子句的最大数量。
值	正整数值。请注意 Oracle Spatial 需要使用小于 1000 的值。
默认值	900

数据绑定

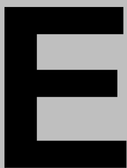
键	com.mapinfo.max.in.items
说明	指定在单一数据绑定 SQL 查询中的 WHERE 子句的 IN 列表中，可以存在的最多元素数量。
值	正整数值。 **Note:** Oracle Spatial 需要使用小于 30000 的值。
默认值	如未指定，则不限制允许使用的元素数量。

字节缓冲

键	com.mapinfo.mibytebuffer
说明	在使用 TAB 库实施时用于执行所有读写的方法。
值	com.mapinfo.util.io.MemMappedByteBuffer － 在使用 TAB 库实施时将使用 java.nio.MappedByteBuffer 来执行所有读写操作。在 Java 中，存在所有映射文件大小总和不能超过 2GB 的限制。 com.mapinfo.util.io.NIOByteBuffer － 在使用 TAB 库实施时将使用 java.nio.FileChannel 来执行所有读写操作。在使用 java.nio.MappedByteBuffer 时，本方法没有大小限制。 **Note:** 通常， com.mapinfo.util.io.MemMappedByteBuffer 执行速度更快，但是其结果可能会有所不同。
默认值	MemMappedByteBuffer

TAB 读取

键	com.mapinfo.tab.read.classname
说明	在执行读取操作时使用其实施。
值	com.mapinfo.tab.dp.TABDataProvider — 将在执行读取 （如 searchAll、searchWithinRectangle 等）时使用 TAB 库实施。 com.mapinfo.dp.tab.ReadOnlyTABDataProvider — 将在执行读取 （如 searchAll、searchWithinRectangle 等）时使用只读实施。在 MapXtreme Java 4.5 和此前版本中均将使用这一实施。 **Note:** 所有更新的方法 （如 updateFeature、 replaceFeature 和 addFeature）均使用新的 TAB 库。
默认值	ReadOnlyTABDataProvider



系统日志记录

本附录介绍在 MapXtreme Java 中可用的日志记录功能。

本附录内容：

日志记录概览

MapXtreme Java 可以记录各种类型的消息，从提供常规信息到严重错误的消息不一而足。记录信息的级别 （重要性）以及位置均可配置。

MapXtreme Java 采用 Jakarta Commons (**http://jakarta.apache.org/commons/logging.html**) 的日志记录 API。借助于这一公共日志记录 API 提供的摘要，可以根据具体需要更改日志记录引擎，而不必更改应用程序代码。

MapXtreme Java 使用 Log4J 作为其日志记录引擎。只要所选日志记录引擎和公共日志记录 API 之间的接口可以通过适配器连接在一起，即可随意更改日志记录程序。有关如何配置 Log4J 的详细信息，请访问 Apache Jarkarta Log4J 站点，网址为 **http://jakarta.apache.org/log4j/docs/index.html**。

在服务器端记录日志

如果使用 Log4J 作为日志记录引擎，则可以根据具体需求，修改随 MapXtreme Java 提供的 log4j.properties 文件。

在客户机端记录日志

如果使用 Log4J 作为日志记录引擎，则可以根据具体需求，修改随 MapXtreme Java 提供的 log4j.properties 文件。

日志记录级别

MapXtreme Java 使用的日志记录级别如下。

级别	说明 / 用途	示例
FATAL（严重）	所发生的意外错误导致系统处于不稳定 / 不可用状态。无法继续执行程序，应用程序必须重新启动。	Servlet 初始化由于无法访问配置文件而失败。
ERROR（错误）	所发生的意外错误导致系统无法完全满足用户请求。程序可以继续执行。	渲染由于命名样式无效而失败或无法完成显示。
WARN（警告）	出现可以 “预见” 的错误，系统可以恢复该错误并完成用户请求。	指定的值无效 － 回退为采用可接受的默认值。
INFO（通知）	一种通知用户信息的方式。	告知用户正在使用的特定系统属性。
DEBUG（调试）	显示有助于证实某些对象正常工作或帮助您解释问题的跟踪信息。	显示在渲染期间，在某个区域的每个点列表上有多少个点。

定制符号

F

本附录展示定制符号的 GIF 图像。

符号

这些定制符号 GIF 图像均位于 MapXtreme-4.7.0\support\symbols\Custom 目录或 MapXtreme-4.7.0\lib\client\previewgifs.jar 文件之内。有关在应用程序中如何使用这些符号的详细信息，请参阅*第 14 章标注和样式*。

AMBU1-32.gif	BADG1-32.gif	BADG2-32.gif	BANK1-32.gif	BANK2-32.gif	BOOK1-32.gif
CAMP1-32.gif	CAR1-32.gif	CAUT1-32.gif	CHUR1-32.gif	COMP1-32.gif	DB-CON.gif
FARM1-32.gif	FAST1-32.gif	FIRE1-32.gif	GLOB1-32.gif	GOLF1-32.gif	HOSP1-32.gif
HOUS1-32.gif	HOUS2-32.gif	HOUS3-32.gif	HYDR1-32.gif	INTE1-32.gif	LITE1-32.gif
LITE2-32.gif	MAIL1-32.gif	MBOX1-32.gif	MBOX2-32.gif	MOSQ1-32.gif	ONEW1-32.gif

ONEW2-32.gif	PENC1-32.gif	PIN1-32.gif	PIN2-32.gif	PIN3-32.gif	PIN4-32.gif
PIN5-32.gif	PIN6-32.gif	POLI1-32.gif	RAIL1-32.gif	RAIL2-32.gif	RAIL3-32.gif
REST1-32.gif	STAT1-32.gif	STOP1-32.gif	SYNA1-32.gif	TARG1-32.gif	TAXI1-32.gif
TEMP1-32.gif	TOWE1-32.gif	TOWE2-32.gif	TRAF1-32.gif	TRUC1-32.gif	TRUC2-32.gif
YIEL1-32.gif	YIEL2-32.gif				

索引

A

B

C

D

E

F

G

H

J

K

L

M

W

X